МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Литвиненко О.Є.

«___» _____2020 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ "МАГІСТР"

Тема: Нейронна система визначення користувацьких переваг

Виконавець: Жмуденко І.Р.

Керівник: Нечипорук В.В.

Нормоконтролер: Тупота Є.В.

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Освітнього ступеня магістр
Спеціальність 123 "Комп'ютерна інженерія"
(шифр, найменування)

Спеціалізація 123.02 "Системне програмування"
(шифр, найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Литвиненко О. Є.

«____»_____2020 р.

ЗАВДАННЯ
на виконання дипломної роботи (проекту)

Жмуденка Івана Романовича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема роботи: Нейронна система визначення користувацьких переваг
затверджена наказом ректора від " 27 " серпня 2020 року № 1203 /ст.
2. Термін виконання роботи: з 05.10.2020 до 31.12.2020
3. Вихідні дані до роботи: нейронна система для реалізації моделі спільної фільтрації
4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):
1) аналіз формальних понять;
2) методи побудови системи визначення користувацьких переваг;
3) проектування нейронної мережі визначення користувацьких переваг.
5. Перелік обов'язкового графічного матеріалу:
1) програмні системи з оцінкою переваг користувачів;
2) перетин безлічі рекомендованих, релевантних і протестованих музичних кліпів;
3) головне вікно сайту з визначенням користувацьких переваг;
4) вікно сайту з запропонованими музичними композиціями;
5) схема алгоритму фільтрації музичних композицій;
6) схема алгоритму рекомендації музичних композицій.

6. Календарний план

№ п/п	Етапи виконання дипломної роботи	Термін виконання етапів	Примітка
1	Ознайомитись з постановкою задачі дипломного проектування	05.10.2020 – 07.10.2020	
2	Вивчити спеціальну літературу і технічну документацію	08.10.2020 – 11.10.2020	
3	Проаналізувати типові моделі спільної фільтрації	12.10.2020 – 15.10.2020	
4	Написати розділ 1.	16.10.2020 – 23.10.2020	
5	Проаналізувати методи і алгоритми роботи системи	24.10.2020 – 29.10.2020	
6	Написати розділ 2.	30.10.2020 – 12.11.2020	
7	Провести опис роботи системи	13.11.2020 – 01.12.2020	
8	Написати розділ 3.	02.12.2020 – 07.12.2020	
9	Оформити пояснювальну записку	08.12.2020 – 13.12.2020	
10	Підготувати презентаційний матеріал	18.12.2020	

7. Дата видачі завдання 05.10.2020

Керівник Нечипорук В.В.
(підпис)

Завдання прийняв до виконання Жмуденко І.Р.
(підпис студента)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи “Нейронна система визначення користувацьких переваг”: 81 с., 30 рис., 22 літературних джерел, 1 додаток.

КОРИСТУВАЦЬКА ПЕРЕВАГА, ГРУПОВА ФІЛЬТРАЦІЯ, НЕЙРОННА МЕРЕЖА, АНАЛІЗ ПОПИТУ, МАШИННЕ НАВЧАННЯ.

Мета дипломної роботи – аналіз методів визначення користувацьких переваг для реалізації програмного забезпечення групової фільтрації.

Об'єкт дослідження – визначення користувацьких переваг.

Предмет дослідження – нейронна система визначення користувацьких переваг.

Наукова значимість полягає у реалізації нейронної мережі визначення користувацьких переваг.

Практична значимість полягає у розробці програмного забезпечення, що планується використовувати у навчальному процесі, як практична складова у вивченні застосування нейронних мереж, та на веб-русурсах для формування переліку пропозицій.

Прогнозні припущення щодо подальшого розвитку матеріалів роботи полягає у навчанні нейронної мережі для покращення результатів групової фільтрації.

Публікації: Нечипорук В.В., Жмуденко І.Р. Нейронна система визначення користувацьких переваг// Тези доповідей наук.-практ. конф. “Сучасні тенденції розвитку системного програмування” (25-26 листопада 2020 р.). – К.: НАУ, 2020. – С. 19.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ANN – *artificial neural networks*

ARM – *advanced risc machine*

ISO – *international organization for standardization*

ПММ – приховані Марковські моделі

ШНМ – штучна нейронна мережа

ВСТУП

Перш за все, необхідно дати визначення рекомендаційної системи. Рекомендаційні системи – це спеціальні програми, головна мета яких полягає у формуванні рекомендацій різних продуктів або сервісів для користувачів на основі їх переваг.

З появою Інтернету сильно зросла кількість інформації, з якої люди щодня стикаються. Це означає, що люди повинні орієнтуватися серед надзвичайно великої кількості доступних альтернатив, коли хочуть щось знайти. Наприклад, від вибору нового мобільного телефону або плеєра до пошуку кінофільму для вечірнього перегляду. З іншого боку виступають власники інтернет-магазинів і сервісів: вони зацікавлені в персональній рекламі і рекомендаціях кожному конкретному користувачеві, тому що такий підхід може істотно збільшити прибуток компаній. Як результат, в останні роки інтерес до розробки та вдосконалення існуючих рекомендаційних систем значно виріс.

На сьогоднішній день існує безліч сайтів, що надають будь-якої контент, наприклад, новини, блоги, музика і кіно. Кожен з них містить величезну кількість інформації, але не вся вона може виявитися цікавою конкретному відвідувачеві сайту. Для підбору контенту, який буде корисний певному користувачеві, використовуються рекомендаційні системи.

На відміну від пошукових систем, щоб отримати відповідь, рекомендаційна система не вимагає чіткого запиту. Користувачеві пропонується оцінити деякі об'єкти з колекції і на підставі його оцінок будуються припущення і повертаються мають найтісніший контакт з ним результати.

Рекомендаційні системи дуже затребувані в даний час, так як значно зменшують час перебування корисної інформації. Оскільки розроблюваний сайт містить музичний контент, то він буде широко затребуваний серед користувачів, які цікавляться музикою і хочуть дізнаватися про невідомих ними раніше виконавців і їх альбомах.

В наші дні рекомендаційні системи вже досить поширені і мають велику кількість застосувань. В першу чергу, рекомендаційні системи використовуються в інтернет-комерції для того, щоб допомогти користувачам вибрати відповідні товари. Такі сервіси збирають інформацію про переваги користувачів і намагаються запропонувати їм корисні товари. Яскравими прикладами компаній, що використовують даний підхід, є *Amazon*, *eBay*, *iTunes* та інші. Інша важлива застосування рекомендаційних систем – це допомога користувачам у виборі музичних композицій, музики і фільмів. Наприклад, сервіси *Pandora*, *GoodReads*, *and IMDb* використовують рекомендаційні системи для цих цілей.

На даний момент існує безліч методів для формування рекомендацій, але всі вони мають свої переваги і недоліки [4], [5]. Саме тому дослідження в даній області актуальні.

На даний момент глибоке навчання користується величезною ажіотажем. Протягом кількох останніх десятиліть спостерігався надзвичайний успіх глибокого навчання (*DL*) у багатьох областях застосування, таких як комп'ютерний зір та розпізнавання мови. Наукові кола та промисловість змагалися застосовувати глибоке навчання для більш широкого кола програм завдяки своїй здатності вирішувати багато складних завдань, забезпечуючи при цьому новітні результати [7]. Нещодавно поглиблене навчання кардинально змінило архітектуру рекомендацій та надало більше можливостей для поліпшення роботи рекомендатора. Нещодавні досягнення в системах рекомендацій на основі глибокого навчання здобули значне значення, подолавши перешкоди звичайних моделей та досягнувши високої якості рекомендацій. Глибоке навчання здатне ефективно фіксувати нелінійні та нетривіальні взаємозв'язки між елементом та елементом та уможливлювати кодування коду складніших абстракцій як подання даних у вищих шарах. Крім того, він вловлює складні взаємозв'язки в самих даних із великої кількості доступних джерел даних, таких як контекстна, текстова та візуальна інформація.

У промисловості системи, що рекомендують, є критично важливими інструментами для покращення взаємодії з користувачем та стимулювання продажів / послуг для багатьох веб-сайтів та мобільних додатків [5].

Наприклад, 80 відсотків фільмів, які дивляться на *Netfix,* походять із рекомендацій [5], 60 відсотків кліків відео надходили з рекомендацій домашньої сторінки на *YouTube* [3]. Останнім часом багато компаній використовують глибоке навчання для подальшого підвищення якості рекомендацій [2, 7, 11]. Ковінгтон та ін. [7] представив глибокий алгоритм рекомендацій на основі нейронної мережі для відеорекомендацій на *YouTube*. *Cheng* та співавт [2] запропонував систему рекомендацій додатків для *Google Play* із широкою та глибокою моделлю. Шумпей та ін. представив систему рекомендацій новин на основі *RNN* для *Yahoo News*. Усі ці моделі витримали онлайн-тестування та продемонстрували суттєве покращення порівняно з традиційними моделями. Тому ми можемо бачити, що глибоке навчання спричинило значну революцію у застосуванні промислових рекомендацій.

Кількість дослідницьких публікацій щодо методів рекомендацій, заснованих на глибокому навчанні, експоненціально збільшилася за ці роки, що наводить вагомі докази неминучої поширеності глибокого навчання у дослідженнях систем, що рекомендують.

Успіх глибокого навчання для рекомендацій як в академічних колах, так і в галузі вимагає всебічного огляду та короткого викладу для послідовних дослідників та практиків, щоб вони розуміли силу та слабкість та сценарії застосування цих моделей.

Було проведено чимало досліджень у галузі рекомендацій на основі глибокого навчання. Однак, наскільки нам відомо, існує дуже мало систематичних оглядів, які добре формують цю сферу та позиціонують існуючі роботи та поточний прогрес. Незважаючи на те, що деякі роботи досліджували програми, що рекомендують, побудовані на методах глибокого навчання, і намагалися формалізувати це дослідження, мало хто намагався надати поглиблений короткий огляд поточних відомостей або деталізувати відкриті проблеми, що існують у цій області. Це опитування має на меті надати такий вичерпний підсумок сучасних досліджень систем рекомендацій на основі глибокого навчання, виявити відкриті проблеми, які в даний час обмежують реалізацію в реальному світі, та вказати майбутні напрямки у цьому вимірі.

Головне завдання цієї роботи – розробка рекомендаційної системи кінофільмів на основі візерункових структур і дослідження її практичної користі. Візерункові структури дозволяють використовувати методи Аналізу Формальних Понять для даних зі складними описами, наприклад інтервалами або графами [2].

Варто відзначити, що Аналіз Формальних Понять вже застосовувався при побудові рекомендаційних систем на основі коллаборативной фільтрації [7]. У тій роботі було розроблено два методи, які використовували грати понять для пошуку «найближчих сусідів» і прогнозування. В іншій роботі використовували візерункові структури для пошуку генів зі схожими біологічними функціями [3].

Мета дипломної роботи – аналіз методів визначення користувацьких переваг для реалізації програмного забезпечення групової фільтрації.

Об'єкт дослідження – визначення користувацьких переваг.

Предмет дослідження – нейронна система визначення користувацьких переваг.

Крім представленого в роботі методу рекомендацій на основі візерункових структур, також було розглянуто і реалізований метод рекомендацій *Slope One* [6]. Алгоритм *Slope One* використовувався для експериментальної перевірки методу, розробленого на основі візерункових структур.

Для експериментів використовувалися вільно поширювані дані оцінок музичних кліпів *AudioMovieLens*, які містять 100000 оцінок від 943 користувачів по тисяча шістсот вісімдесят двом музичним кліпамв.

Експериментальне порівняння розробленого алгоритму і алгоритму *Slope One* проводилося для різних конфігурацій за трьома критеріями: середній час роботи, середня точність і середня повнота.

Для проведення експериментів було написано програму для побудови нейронної мережі:

– програма, яка обробляє початкові дані і перетворює їх в глобальні матриці, з якими працюють всі інші програми.

– програма, яка запускає методи з різними параметрами і порівнює отримані результати в термінах точності і повноти.

– програма, яка реалізує алгоритм рекомендацій на основі візерункових структур.

– програма, яка реалізовувала алгоритм *Slope One*.

– програма для обчислення точності і повноти.

Результати експериментів показали, що розроблений метод рекомендацій на основі візерункових структур дає непогані рекомендації і застосуємо для практичного використання.

Дана робота розбита на три смислові частини. У першій розглядаються основні визначення Аналізу Формальних Понять і візерункових структур. У другій частині, розглядаються існуючі підходи до рекомендаційних систем, наводяться алгоритм *Slope One* і алгоритм рекомендацій на основі візерункових структур з прикладами застосування. Заключна частина присвячена проведеним експериментам і всьому з ними пов'язаного.

РОЗДІЛ 1
АНАЛІЗ ФОРМАЛЬНИХ ПОНЯТЬ

1.1. Основні визначення

Базові визначення аналізу формальних понять використовуємо з [1].

Нехай і – деякі множини, а – бінарне відношення між і, таке що, тобто – підмножина їх декартового твори. Тоді трійка називається формальним контекстом. Елементи безлічі прийнято називати об'єктами, а елементи множини називають ознаками формального контексту Попросту кажучи, пара (або) означає, що і знаходяться в бінарному відношенні і це можна інтерпретувати як "об'єкт має ознаку". Два оператора, визначені нижче, задають відповідність Галуа:

$$GMIGMI \subseteq G \times M(G,M,I)G(g \in G)M(m \in M)(G,M,I).(g,m) \in IgImgmIgm(.)'(2^G,\subseteq)(2^M,\subseteq)$$

$$A' = \{m \in M | \forall g \in A: gIm\} \text{ для } A \subseteq G \quad (1.1)$$

$$B' = \{g \in G | \forall m \in B: gIm\} \text{ для } B \subseteq M \quad (1.2)$$

Оператор (1.1) для деякого безлічі об'єктів повертає максимальне безліч ознак, якими володіє кожен об'єкт з безлічі. Аналогічним чином, оператор (1.2) для деякого безлічі ознак повертає максимальне безліч об'єктів, які мають всі ознаки з. $AABB$

Тепер визначимо формальне поняття. Пара, де і, називається формальним поняттям, якщо вона задовольняє умовам: і. У цьому випадку безліч об'єктів називається об'ємом формального поняття, а безліч ознак називається змістом формального поняття.

$$(A,B)\text{A} \subseteq \text{GB} \subseteq \text{M}A' = BB' = AA(A,B)B(A,B)$$

Формальні поняття зазвичай частково впорядковані:. Для даного часткового порядку, безліч всіх частково впорядкованих формальних понять позначається як $\mathscr{B}$ і називається гратами понять формального контексту.

$$(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2 (\Leftrightarrow B_2 \subseteq B_1)(G, M, I)(G, M, I)$$

1.2. Візерункові структури

1.2.1. Введення в візерункові структури

Нехай – деяка множина об'єктів, – множина всіх можливих візерунків (*patterns*), які також називають описами об'єктів. Нехай – операція подібності (*similarity operator*). Ця операція дозволяє працювати з об'єктами, які в якості ознак мають не бінарні ознаки, як в класичному аналізі формальних понять, а ознаки зі складним описом, наприклад інтервалом або графом. Тоді полурешетка перетинів (*meet-semi-lattice*) описів об'єктів. Відображення зіставляє об'єкту опис.

$$GD \sqcap (D,\sqcap)\delta: G \to Dgd \in (D,\sqcap)$$

Трійку будемо називати візерункової структурою. Два оператора визначені нижче задають відповідність Галуа між і:$(G, (D,\sqcap), \delta)(\,.\,)^{\square}(2^G, \subseteq)(D,\sqcap)$

$$A^{\square} = \sqcap_{g \in A} \delta(g) \quad \text{для } A \subseteq G \tag{1.3}$$

$$d^{\square} = \{g \in G | d \sqsubseteq \delta(g)\} \quad \text{для } d \in (D,\sqcap), \text{где}$$

$$d \sqsubseteq \delta(g) \Leftrightarrow d \sqcap \delta(g) = d \tag{1.4}$$

Оператор (1.3) будемо називати першим оператором Галуа (*derivation operator*). Він для безлічі об'єктів повертає загальне максимальне опис (візерунок) всіх об'єктів з. Оператор (1.4) будемо називати другим оператором Галуа. Він для опису повертає безліч всіх об'єктів, які поділяють опис.

Пара, де і, називається візерунковим поняттям візерункової структури, якщо вона задовольняє двом умовам: і. У цьому випадку безліч об'єктів називається об'ємом візерункового поняття (*A*, *d*), а опис називається змістом візерункового поняття.

$$(A,d)\mathrm{A} \subseteq \mathrm{G}d \in (D,\sqcap)(G,(D,\sqcap),\delta)A^{\square} = dd^{\square} = AAd(A,d)$$

Візерункові поняття частково впорядковані. Для даного часткового порядку, безліч всіх частково впорядкованих візерункових понять формує грати візерункових понять візерункової структури.

$$(A_1,d_1) \leq (A_2,d_2) \Leftrightarrow A_1 \subseteq A_2(\Leftrightarrow d_2 \sqsubseteq d_1)(G,(D,\sqcap),\delta)$$

1.2.2. Інтервали в якості візерунків

Очевидно, що реалізація операції подібності між двома інтервалами повинна задовольняти такій умові, що ці два інтервали повинні лежати в деякому новому інтервалі, який містить вихідні. Покладемо цей новий інтервал буде мінімальним інтервалом, який містить два вихідних. Якщо дані два інтервали і такі, що, і, тоді визначимо операцію подібності двох інтервалів як:

$$\begin{gathered}[a_1,b_1][a_2,b_2]a_1,b_1,a_2,b_2 \in \mathbb{R}a_1 \leq b_1 a_2 \leq b_2 \\ [a_1,b_1] \sqcap [a_2,b_2] = [\min(a_1,a_2),\max(b_1,b_2)]\end{gathered} \tag{1.5}$$

Звідки очевидно, що:

$$\begin{gathered}[a_1,b_1] \sqsubseteq [a_2,b_2] \Leftrightarrow [a_1,b_1] \sqcap [a_2,b_2] = [a_1,b_1] \\ \Leftrightarrow [\min(a_1,a_2),\max(b_1,b_2)] = [a_1,b_1] \\ \Leftrightarrow a_1 \leq a_2 \ \text{и} \ b_1 \geq b_2 \Leftrightarrow [a_1,b_1] \supseteq [a_2,b_2]\end{gathered} \tag{1.6}$$

Також необхідно зробити зауваження про те, що дійсне число можна представити інтервалом $a \in \mathbb{R}[a,a]$.

1.2.3. Інтервальні вектори в якості візерунків

Будемо називати *p*-розмірні вектори інтервалів інтервальними векторами. В даному випадку для інтервальних векторів однакової розмірності і визначимо операцію подібності як перетин відповідних координат інтервальних векторів, тобто:

$$e = \langle [a_i, b_i] \rangle_{i \in [1,p]} f = \langle [c_i, d_i] \rangle_{i \in [1,p]}$$
$$e \sqcap f = \langle [a_i, b_i] \rangle_{i \in [1,p]} \sqcap \langle [c_i, d_i] \rangle_{i \in [1,p]} \Leftrightarrow e \sqcap f = \langle [a_i, b_i] \sqcap [c_i, d_i] \rangle_{i \in [1,p]} \quad (1.7)$$

Слід зауважити, що інтервальні вектори можуть бути також частково впорядковані:

$$e \sqsubseteq f \Leftrightarrow \langle [a_i, b_i] \rangle_{i \in [1,p]} \sqsubseteq \langle [c_i, d_i] \rangle_{i \in [1,p]}$$
$$\Leftrightarrow [a_i, b_i] \sqsubseteq [c_i, d_i] \text{ для всех } i \in [1, p] \quad (1.8)$$

1.3. Поняття рекомендаційних системи

Існує великий клас веб-додатків, які передбачають прогнозування відповідей користувачів на варіанти. Такий об'єкт називається *а*система рекомендацій. почнемо цю главу з огляду найважливіших прикладів цих систем. Однак, щоб привернути увагу до проблеми, є два прикладні приклади систем рекомендацій:

1. Пропонування статей новин для читачів газет на основі прогнозування інтересів читачів.

2. Пропонування клієнтам он-лайн роздрібної торгівлі пропозиції щодо того, що вони можуть придбати, виходячи з їхньої минулої історії покупок та / або пошуку товарів.

Системи рекомендацій використовують низку різних технологій. можемо класифікувати ці системи на дві широкі групи.

– системи на основі вмісту вивчити властивості рекомендованих предметів. Для наприклад, якщо користувач *Netflix* дивився багато ковбойських фільмів, то рекомендуйте фільм, класифікований у базі даних як такий, що має жанр "ковбой".

– спільна фільтрація системи рекомендують предмети на основі вимірів схожості між користувачами та / або предметами. Елементи, рекомендовані користувачеві, – це ті, що віддають перевагу подібні користувачі. Така система

рекомендацій може використовувати основи, викладені в главі 3 щодо пошуку подібностей та главі 7 щодо кластеризації. Однак ці технології самі по собі не є достатніми, і є деякі нові алгоритми, які довели свою ефективність для систем рекомендацій.

1.3.1. Модель систем рекомендацій

Введемо поняття "довгий хвіст", що пояснює перевагу он-лайн продавців перед звичайними продавцями цегли та розчину. Потім коротко оглядаємо типи програм, в яких системи рекомендацій виявилися корисними.

У додатку системи рекомендацій є два класи сутностей, які будемо називати користувачі і предмети. Користувачі мають переваги щодо певних предметів, і ці уподобання повинні бути вилучені з даних. Самі дані представлені якматриця корисності, даючи для кожної пари елемент-користувач значення, яке представляє те, що відомо про ступінь переваги цього користувача щодо цього елемента. Значення походять із упорядкованого набору, наприклад, цілих чисел 1–5, що представляють кількість зірочок, які користувач дав як оцінку цьому елементу. вважаємо, що матриця розріджена, тобто більшість записів є "невідомими". Невідомий рейтинг означає, що не маємо явної інформації про переваги користувача щодо товару.

Не потрібно передбачати кожен порожній запис у матриці утиліти. Навпаки, потрібно лише виявити деякі записи в кожному рядку, які, ймовірно, будуть високими. У більшості додатків система рекомендацій не пропонує користувачам ранжирування всіх елементів, а навпаки, пропонує кілька, які користувач повинен високо оцінити. Можливо, навіть не потрібно буде знаходити всі елементи з найвищими очікуваними оцінками, а лише знаходити велику підмножину з найвищими оцінками.

Довгий хвіст – явище, яке робить системи рекомендацій необхідними. Фізичні системи доставки характеризуються дефіцитом ресурсів. Цегляно-будівельні магазини мають обмежений простір на полицях і можуть показати покупцеві лише незначну частину всіх варіантів, що існують. З іншого боку, он-лайн магазини можуть зробити все, що існує, доступним для клієнта. Таким

чином, у фізичній книгарні на полицях може бути кілька тисяч книг, але *Amazon* пропонує мільйони книг. Фізична газета може надрукувати кілька десятків статей на день, тоді як онлайнові служби новин пропонують тисячі на день.

Рекомендації у фізичному світі досить прості. По-перше, неможливо пристосувати магазин до кожного окремого покупця. Таким чином, вибір того, що надається, регулюється лише сукупними числами. Як правило, в книгарні відображатимуться лише ті книги, які є найбільш популярними, а газета друкуватиме лише ті статті, які, на її думку, зацікавлять найбільше людей. У першому випадку показники продажів визначають вибір, у другому випадку редакційне рішення подає.

Вироби впорядковуються на горизонтальній осі відповідно до їх популярності. Фізичні установи надають лише найпопулярніші предмети ліворуч від вертикальної лінії, тоді як відповідні он-лайн заклади надають весь спектр предметів: хвіст, а також популярні предмети.

Явище довгих хвостів змушує онлайнові установи рекомендувати товари окремим користувачам. Неможливо представити користувачеві всі доступні предмети, як це можуть фізичні установи. Також не можемо очікувати, що користувачі почують про кожен з предметів, який може їм сподобатися.

1.3.2. Застосування систем рекомендацій

Наведемо список основних можлививх.

1. Рекомендації щодо продуктів: Мабуть, найважливіше використання систем рекомендацій – це інтернет-магазини. зазначали, як *Amazon* або подібні он-лайн постачальники прагнуть представити кожному користувачеві, що повертаються, пропозиції щодо товарів, які вони могли б придбати. Ці пропозиції не є випадковими, а базуються на рішеннях про закупівлю, прийнятих подібними замовниками, або на інших методах, про які поговоримо в цій главі.

2. Кінорекомендації: *Netflix* пропонує рекомендації своїм клієнтам фільмів, які можуть їм сподобатися. Ці рекомендації базуються на рейтингах, наданих користувачами, подібно до рейтингів. Важливість точного прогнозування рейтингів полягає настільки високий, що *Netflix* отримав приз у мільйон доларів

за перший алгоритм, який міг би перевершити власну систему рекомендацій на 10%.1 Нарешті приз був виграний у 2009 році групою дослідників під назвою «Прагматичний хаос Беллкора», після більш ніж трьох років змагань.

3. Статті новин: служби новин намагалися ідентифікувати статті, що цікавлять читачів, на основі статей, які вони прочитали в минулому. Подібність може базуватися на схожості важливих слів у документах або на статтях, які читають люди зі схожими смаками читання. Ті самі принципи застосовуються до рекомендацій щоденників із мільйонів доступних щоденників, відео на *YouTube* чи інших сайтах, де контент регулярно надається.

1.3.3. Заповнення матриці службових програм

Без матриці корисних програм практично неможливо рекомендувати елементи. Однак отримання даних, з яких можна побудувати матрицю корисності, часто є складним. Існує два загальних підходи до виявлення цінності, яку користувачі надають елементам.

1. Можемо попросити користувачів оцінити елементи. Рейтинг фільмів зазвичай отримується таким чином, і деякі інтернет-магазини намагаються отримати рейтинги у своїх покупців. Сайти, що надають вміст, такі як деякі новинні сайти або *YouTube*, також просять користувачів оцінювати елементи. Цей підхід обмежений своєю ефективністю, оскільки, як правило, користувачі не бажають давати відповіді, а інформація тих, хто робить, може бути упередженою саме тим фактом, що вона надходить від людей, які бажають надати рейтинги.

2. Можемо робити висновки щодо поведінки користувачів. Найбільш очевидно, що якщо користувач купує продукт на *Amazon*, дивиться фільм на *YouTube* або читає новинну статтю, тоді користувачеві можна сказати, що він подобається цьому товару. Зверніть увагу, що така система рейтингу насправді має лише одне значення: 1 означає, що продукт сподобався користувачеві. Часто знаходимо матрицю утиліти з даними такого типу, що відображаються цифрами 0, а не порожніми місцями, коли користувач не придбав або не переглянути товар. Однак у цьому випадку 0 – це не нижчий рейтинг, ніж 1; це зовсім не рейтинг. Більш загально, можна зробити висновок про інтерес з поведінки,

відмінної від покупки. Наприклад, якщо клієнт *Amazon* переглядає інформацію про товар, можемо зробити висновок, що його цікавить товар, навіть якщо він його не купує.

1.3.4. Рекомендації на основі вмісту

Існує дві основні архітектури системи рекомендацій:

– на основі вмісту системи орієнтуються на властивості предметів. Подібність предметів визначається шляхом вимірювання подібності за їх властивостями;

– спільне фільтрування системи орієнтуються на взаємини між користувачами та предмети. Схожість предметів визначається схожістю оцінок цих предметів користувачами, які оцінили обидва елементи.

У даний роботі зосереджуємось на системах рекомендацій на основі вмісту. У системі, що базується на вмісті, повинні побудувати для кожного елемента *a* профіль, який є записом або колекцією записів, що представляють важливі характеристики цього елемента. У простих випадках профіль складається з деяких характеристик предмета, які легко виявити. Наприклад, розглянемо особливості фільму, які можуть мати відношення до системи рекомендацій:

– набір акторів фільму. Деякі глядачі віддають перевагу фільмам зі своїми улюбленими акторами;

– режисер. Деякі глядачі віддають перевагу роботі певних режисерів;

– рік, в якому був знятий фільм. Деякі глядачі віддають перевагу старим фільмам; інші дивляться лише останні випуски.

– жанр або загальний тип фільму. Одні глядачі люблять лише комедії, інші драми чи романси.

Є багато інших особливостей фільмів, які також можна використовувати. За винятком останнього жанру, інформація легко доступна з описів фільмів. Жанр – це неясне поняття. Однак огляди фільмів зазвичай визначають жанр із набору загальновживаних термінів. НаприкладІнтернет-фільм База даних (*IMDB*) призначає жанр або жанри кожному фільму.

Багато інших класів предметів також дозволяють отримувати функції з наявних даних, навіть якщо ці дані в певний момент повинні бути введені вручну. Наприклад, продукти часто мають описи, написані виробником, із зазначенням характеристик, що відповідають даному класу товарів (наприклад, розмір екрану та колір шафи для телевізора). Книги мають описи, подібні до описів для фільмів, тому можемо отримати такі функції, як автор, рік видання та жанр. Музичні продукти, такі як завантаження компакт-дисків та *MP*3, мають такі функції, як виконавець, композитор та жанр.

Існують інші класи предметів, де не одразу стає очевидним, якими мають бути значення ознак. розглянемо два з них: колекції документів та зображення. Документи представляють особливі проблеми, і в цьому розділі обговоримо технологію вилучення об'єктів з документів.

Існує багато видів документів, для яких система рекомендацій може бути корисною. Наприклад, щодня публікується багато статей новин, і не можемо прочитати їх усіх. Система рекомендацій може пропонувати статті на теми, які цікавлять користувача, але як можемо розрізняти теми? Веб-сторінки – це також колекція документів. Чи можемо запропонувати сторінки, які користувач може побачити? Так само блоги можна рекомендувати зацікавленим користувачам, якби могли класифікувати блоги за темами.

На жаль, ці класи документів, як правило, не мають легко доступних функцій, що надають інформацію. Замінником, який був корисний на практиці, є визначення слів, що характеризують тему документа. По-перше, усуньте стоп-слова – кілька сотень найпоширеніших слів, які, як правило, мало говорять про тему документа. Для решти слів обчисліть оцінку *TF.IDF* для кожного слова в документі. Найвищі бали – це слова, що характеризують документ.

Тоді можемо взяти за ознаки документа *n* слів із найвищими балами *TF.IDF*. Можна вибрати *n* однаковим для всіх документів або дозволити *n* фіксованим відсотком слів у документі. також можемо вибрати, щоб усі слова, оцінки *TF.IDF* яких перевищують заданий поріг, були частиною набору функцій.

Тепер документи представлені наборами слів. Інтуїтивно очікуємо, що ці слова виражають тему або основні ідеї документа. Наприклад, у статті новин

очікуємо, що слова з найвищим балом *TF_IDF* включатимуть імена людей, про яких йдеться у статті, незвичні властивості описаної події та місце події. Щоб виміряти схожість двох документів, можемо використати кілька природних вимірювань відстані:

Розглянемо базу даних зображень, як приклад способу отримання характеристик для предметів. Проблема зображень полягає в тому, що їхні дані, як правило, масив пікселів, не говорять нічого корисного про їх функції. можемо розрахувати прості властивості пікселів, такі як середня кількість червоного на зображенні, але небагато користувачів шукають червоні картинки або особливо подобаються червоні картинки.

Було здійснено ряд спроб отримати інформацію про особливості предметів, запросивши користувачів тегпункти шляхом введення слів або фраз, що описують елемент. Таким чином, на одній картині з великою кількістю червоного може бути позначено "Площа Тяньань-мен", а на іншій – "захід сонця в Малібу". Різниця не є тим, що можна виявити за допомогою існуючих програм аналізу зображень.

Майже будь-який тип даних може мати свої функції, описані тегами. Однією з перших спроб позначити величезний обсяг даних був сайт *del.icio.us*, який пізніше придбав *Yahoo*!, який запрошував користувачів позначати веб-сторінки. Метою цього позначення було зробити доступним новий метод пошуку, коли користувачі вводили набір тегів як свій пошуковий запит, а система отримувала веб-сторінки, які були позначені таким чином. Однак можна також використовувати теги як систему рекомендацій. Якщо спостерігається, що користувач отримує або додає до закладок багато сторінок із певним набором тегів, тоді можемо рекомендувати інші сторінки з тими самими тегами.

Проблема тегування як підходу до виявлення особливостей полягає в тому, що процес працює лише в тому випадку, якщо користувачі готові взяти на себе зусилля, щоб створити теги, і існує достатня кількість тегів, які випадкові помилкові не будуть занадто упереджувати систему.

Кінцевою метою рекомендацій на основі вмісту є створення як елемента профілю, що складається з пар об'єкт-значення, так і профілю користувача, що узагальнює переваги користувача на основі їхнього рядка матриці утиліти.

Узагальнимо цей векторний підхід до всіляких ознак. Це легко зробити для функцій, що являють собою набори дискретних значень. Наприклад, якщо однією особливістю фільмів є безліч акторів, то уявіть, що для кожного актора є компонент: 1, якщо актор у фільмі, і 0, якщо ні. Так само можемо скласти компонент для кожного можливого режисера та кожного можливого жанру. Усі ці функції можна представити, використовуючи лише 0 і 1.

Існує ще один клас функцій, який не буває легко представити булевими векторами: ті об'єкти, які є числовими. Наприклад, можемо прийняти середній рейтинг для фільмів як особливість, і це середнє значення є дійсним числом. Немає сенсу мати один компонент для кожного з можливих середніх рейтингів, і це призведе до втрати структури, неявної в числах. Тобто два рейтинги, які є близькими, але не ідентичними, слід вважати більш подібними, ніж широко різняться рейтинги. Подібним чином, числові характеристики продуктів, такі як розмір екрану або ємність диска для ПК, слід вважати подібними, якщо їх значення не сильно відрізняються.

Числові ознаки повинні бути представлені окремими компонентами векторів, що представляють елементи. Ці компоненти містять точне значення цієї функції.

Немає шкоди, якщо деякі компоненти векторів є булевими, а інші – дійсними або цілочисельними. все ще можемо обчислити косинусну відстань між векторами, хоча, якщо це зробимо, слід подумати про відповідне масштабування небулевих компонентів, щоб вони ні домінували в обчисленні, ні вони не мають значення.

Потрібно не лише створювати вектори, що описують предмети; потрібно створити вектори з однаковими компонентами, що описують уподобання користувача. Є матриця утиліт, що представляє зв'язок між користувачами та елементами. Згадаймо, що непусті записи матриці можуть бути лише одиницями, що представляють покупки користувача або подібне з'єднання, або можуть бути

довільними числами, що представляють рейтинг або ступінь впливу, який користувач має на товар.

Маючи цю інформацію, найкращою оцінкою щодо того, які елементи сподобалися користувачеві, є деяке зведення профілів цих предметів. Якщо матриця корисності має лише одиниці, тоді природний сукупний показник є середнім значенням компонентів

За допомогою векторів профілю, як для користувачів, так і для елементів, можемо оцінити ступінь, до якого користувач віддасть перевагу елементу, обчислюючи косинусну відстань між векторами користувача та елемента. Методи випадкового гіперплощинного та місцевого хешування можуть бути використані для розміщення (лише) профілів елементів у відрах. Таким чином, даючи користувачеві, якому хочемо порекомендувати деякі елементи, можемо застосувати однакові дві техніки – випадкові гіперплощини та *LSH* – для визначення того, в яких сегментах повинні шукати елементи, які можуть мати невелику косинусну відстань від користувача.

1.4. Аналіз алгоритмів класифікації

Повністю різний підхід до системи рекомендацій, що використовує профілі елементів та матриці корисності, полягає в тому, щоб розглядати проблему як машинне навчання. Розгляньте подані дані як навчальний набір, і для кожного користувача складіть класифікатор, який передбачає рейтинг усіх предметів. Існує велика кількість різних класифікаторів, і не маємо на меті викладати цю тему тут. Однак ви повинні знати про можливість розробки класифікатора для рекомендацій, тому коротко обговоримо один загальний класифікатор – дерева рішень.

А дерево рішень являє собою сукупність вузлів, розташованих у вигляді двійкового дерева. Листя приймають рішення; у нашому випадку рішення буде “подобається” або “не подобається”. Кожен внутрішній вузол є умовою щодо об’єктів, що класифікуються; у нашому випадку умовою буде предикат, що включає одну або кілька особливостей предмета.

Щоб класифікувати елемент, починаємо з кореня, і застосовуємо предикат у корені до елемента. Якщо предикат істинний, перейдіть до лівої дитини, а якщо вона неправдива, то до правої дитини. Потім повторіть той самий процес у відвіданому вузлі, поки не буде досягнуто лист. Цей аркуш класифікує товар як подобається чи ні.

Побудова дерева рішень вимагає вибору предиката для кожного внутрішнього вузла. Існує багато способів вибрати найкращий присудок, але всі вони намагаються домовитись, щоб хтось із дітей отримав усі або більшість позитивних прикладів у навчальному наборі (тобто елементи, які подобаються даному користувачеві, у нашому випадку) та інша дитина отримує всі або більшість негативних прикладів (предмети, які не подобаються цьому користувачеві).

Після того, як вибрали предикат для вузла N, поділяємо елементи на дві групи: ті, які задовольняють предикату, і ті, які цього не роблять. Для кожної групи знову знаходимо присудок, який найкраще відокремлює позитивні та негативні приклади в цій групі. Ці присудки присвоюються дочірній системі N. Цей процес поділу прикладів та побудови дітей може переходити на будь-яку кількість рівнів. можемо зупинитися і створити аркуш, якщо група елементів для вузла однорідна; тобто всі вони є позитивними чи негативними прикладами.

Однак можемо зупинитися і створити лист із рішенням більшості для групи, навіть якщо група містить як позитивні, так і негативні приклади. Причина полягає в тому, що статистична значимість невеликої групи може бути недостатньо високою, на яку можна було б покластися. З цієї причини варіантною стратегією є створенняансамблюдерев рішень, кожен з яких використовує різні предикати, але дозволяє деревам бути глибшими, ніж обґрунтовують наявні дані. Такі дерева називаютьсяпереобладнаний. Щоб класифікувати предмет, застосуйте всі дерева ансамблю і дайте їм проголосувати за результат. не будемо тут розглядати цей варіант, а наведемо простий гіпотетичний приклад дерева рішень.

Класифікатори всіх типів, як правило, займають багато часу для побудови. Наприклад, якщо хочемо використовувати дерева рішень, потрібно одне дерево на кожного користувача.

1.5. Аналіз методів спільної фільтрації

Будемо застосовувати значно різний підхід до рекомендацій. Замість того, щоб використовувати особливості елементів для визначення їх подібності, зосереджуємось на подібності оцінок користувачів для двох предметів. Тобто замість вектора елемента-профілю для елемента використовуємо його стовпець у матриці утиліти. Далі, замість того, щоб створювати вектор профілю для користувачів, представляємо їх за їх рядками в матриці утиліти. Користувачі подібні, якщо їх вектори близькі за деякою мірою відстані, такою як відстань Жакара або косинуса. Потім рекомендується користувачеві *U*, переглядаючи користувачів, найбільш схожих на *U* в цьому сенсі, та рекомендуючи елементи, які подобаються цим користувачам. Процес ідентифікації схожих користувачів та рекомендації, як подібні користувачі, називаєтьсяспільний фільтрування.

1.5.1. Вимірювання подібності

Перше питання, з яким повинні мати справу, – це як виміряти схожість користувачів або елементів з їх рядків або стовпців у матриці утиліти. Особливо спостерігайте за користувачами A та C. Вони оцінили два спільні фільми, але, схоже, вони мають майже діаметрально протилежні думки щодо цих фільмів. могли б очікувати, що хороший показник відстані зробить їх досить далекими. Ось кілька альтернативних заходів, які слід розглянути.

Якщо нормалізуємо рейтинги, віднімаючи від кожного рейтингу середній рейтинг цього користувача, перетворюємо низькі оцінки в негативні, а високі – в позитивні. Якщо тоді взяти косинусну відстань, виявимо, що користувачі з протилежними поглядами на спільно переглянуті фільми будуть мати вектори майже в протилежних напрямках, і їх можна розглядати якомога далі. Однак

користувачі, які мають подібні думки щодо спільних фільмів, мають відносно невеликий кут між ними.

1.5.2. Аналіз оцінки подвійності подібності

Матрицю корисності можна розглядати як повідомлення про користувачів або про елементи, або про те, і про те. Важливо усвідомити, що будь-який із запропонованих методів пошуку подібних користувачів може бути використаний у стовпцях матриці службових програм для пошуку подібних елементів. Існує два способи порушення симетрії на практиці.

1. Є можливість використовувати інформацію про користувачів, щоб рекомендувати товари. Тобто, даючи користувачеві, можемо знайти деяку кількість найбільш схожих користувачів. Можемо базувати рекомендацію на рішеннях, прийнятих подібними користувачами, наприклад, рекомендувати елементи, які найбільше придбали або оцінили високо. Однак симетрії немає. Навіть якщо знаходимо пари подібних предметів, потрібно зробити додатковий крок, щоб рекомендувати товари користувачам. Цей пункт досліджується далі в кінці цього підрозділу.

2. Існує різниця у типовій поведінці користувачів та предметів, оскільки це стосується подібності. Інтуїтивно зрозуміло, що предмети класифікуються простими термінами. Наприклад, музика, як правило, належить до одного жанру. Неможливо, наприклад, щоб музичний твір був як роком 60-х, так і 1700-х років бароко. З іншого боку, є люди, яким подобається як рок 60-х, так і бароко 1700-х років, і які купують зразки обох типів музики. Наслідком цього є те, що легше виявити подібні предмети, оскільки вони належать до одного жанру, ніж виявити, що два користувачі схожі, тому що вони віддають перевагу одному спільному жанру, тоді як кожному також подобаються деякі жанри, а інші – ні. т догляд за.

Одним із способів прогнозування значення запису матриці корисності для користувача *U* та елемента *I* є пошук *n* користувачів (для деяких заздалегідь визначених *n*), найбільш подібних до *U*, і усереднення їх рейтингу для елемента *I*, враховуючи лише тих серед російських подібних користувачів, які оцінили *I*.

Зазвичай краще спочатку нормалізувати матрицю. Тобто для кожного з російських користувачів віднімітьїх середній рейтинг для предметів з їхнього рейтингу для *i*. Середнє значення різниці для тих користувачів, які оцінили *I*, а потім додайте це середнє значення до середньої оцінки, яку *U* дає для всіх елементів. Ця нормалізація коригує оцінку в тому випадку, якщо *U*, як правило, дає дуже високі або дуже низькі оцінки, або велика частка подібних користувачів, які оцінили *I* (яких може бути лише декілька), є користувачами, які мають тенденцію оцінювати дуже високу або дуже низько.

Вдвічі можемо використовувати схожість елементів для оцінки запису для користувача *U* та елемента *I*. Знайдіть *m* елементів, найбільш схожих на *I*, для деяких *m*, і візьміть середню оцінку серед *m* елементів оцінок, які дав *U*. Що стосується подібності між користувачем та користувачем, розглядаємо лише ті елементи серед *m*, які оцінив *U*, і спочатку нормалізувати рейтинги елементів.

Який би підхід до оцінки записів у матриці корисності не використовували, недостатньо знайти лише один запис. Для того, щоб рекомендувати елементи користувачеві *U*, потрібно оцінити кожен запис у рядку матриці утиліти для *U* або, принаймні, знайти всі або більшість записів у цьому рядку, які є порожніми, але мають високе розрахункове значення. Існує заміна щодо того, чи слід працювати з подібними користувачами чи подібними предметами:

– Якщо знаходимо подібних користувачів, то потрібно виконати процес лише один раз для користувача *U*. З набору подібних користувачів можемо оцінити всі прогалини в матриці утиліти для *U*. Якщо працюємо з подібних елементів, повинні обчислити подібні елементи майже для всіх елементів, перш ніж зможемо оцінити рядок для *U*.

– З іншого боку, подібність предметів-предметів часто забезпечує більш надійну інформацію через явище, яке спостерігалося вище, а саме те, що легше знайти предмети одного жанру, ніж знайти користувачів, яким подобаються лише предмети одного жанру .

Який би метод не вибрали, слід попередньо обчислити бажані елементи для кожного користувача, а не чекати, поки потрібно буде прийняти рішення. Оскільки матриця корисності розвивається повільно, як правило, достатньо її

обчислити нечасто і припустити, що вона залишається фіксованою між перерахунками.

1.5.3. Кластеризація користувачів та предметів

Важко виявити подібність як між елементами, так і серед користувачів, оскільки маємо мало інформації про пари користувач-елемент у розрідженій матриці утиліти. Навіть якщо два елементи належать до одного жанру, найімовірніше буде дуже мало користувачів, які придбали або оцінили обидва. Так само, навіть якщо двом користувачам обом подобається жанр або жанри, вони, можливо, не купували жодних спільних предметів.

Одним із способів вирішення цієї проблеми є кластеризація елементів та / або користувачів. Виберіть будь-яку міру відстані або будь-яку іншу міру відстані, і використовуйте її для групування, скажімо, предметів. Можна використовувати будь-який із методів. Однак побачимо, що може бути мало причин намагатися негайно сгрупуватися в невелику кількість скупчень. Швидше, ієрархічний підхід, коли залишаємо багато скупчень нез'єднаними, може бути достатнім як перший крок. Наприклад, можемо залишити вдвічі менше кластерів, скільки елементів.

.Розмістивши кластеризовані елементи до певної міри, можемо переглянути матрицю утиліти, щоб стовпці представляли кластери елементів, а запис для користувача *U* та кластера *C* – це середня оцінка, яку *U* дав тим членам кластеру *C*, які *U* оцінив. Зверніть увагу, що *U*, можливо, не оцінив жодного з членів кластера, і в цьому випадку запис для *U* та *C* все ще пустий.

Є можливість використовувати цю переглянуту матрицю корисних програм для кластеризації користувачів, знову використовуючи міру дистанції, яку вважаємо найбільш доцільною. Використовуйте алгоритм кластеризації, який знову залишає багато кластерів, наприклад, вдвічі менше кластерів, ніж кількість користувачів. Повторно візуйте матрицю утиліти, щоб рядки відповідали кластерам користувачів, так само, як стовпці відповідають кластерам елементів. Що стосується кластерів елементів, обчисліть запис для кластера користувачів, усереднюючи рейтинги користувачів у кластері.

Тепер цей процес можна повторити кілька разів, якщо хочемо. Тобто можемо кластеризувати кластери елементів і знову об’єднати стовпці матриці службових програм, які належать до одного кластера. Потім можемо знову звернутися до користувачів і кластеризувати їх. Процес може повторюватися, поки не отримаємо інтуїтивно розумну кількість кластерів кожного виду.

Після того, як кластерували користувачів та / або елементи до бажаного обсягу та обчислили матрицю утиліти кластер-кластер, можемо оцінювати записи в початковій матриці утиліти наступним чином. Припустимо, хочемо передачити запис для користувача *U* та елемента *I*:

1.6. Висновки до розділу

У цьому розділі були розглянуті основні визначення аналізу формальних понять і візерункових структур, які використовуються для класифікації груп користувачів за їх вподобаннями. Також були розглянуті розширення візерункових структур для інтервалів і інтервальних векторів.

РОЗДІЛ 2
МЕТОДИ ПОБУДОВИ СИСТЕМИ ВИЗНАЧЕННЯ КОРИСТУВАЦЬКИХ ПЕРЕВАГ

2.1. Аналіз існуючих підходів до визначення користувацьких переваг

На сьогоднішній день при створенні рекомендаційних систем використовуються дві основні стратегії: фільтрація вмісту і коллаборативна фільтрація. Відразу варто відзначити, що на практиці зазвичай використовуються гібридні методи, що поєднують в собі переваги розглянутих нижче підходів.

2.1.1. Визначення користувацьких переваг через фільтрація вмісту

Фільтрація вмісту вимагає створення профілів для користувачів і об'єктів, які будуть рекомендуватися.

Профіль об'єкта може містити сотні ознак, що відображають найбільш повно його властивості. Причому профілі об'єктів заповнюються експертами в конкретній галузі. Наприклад, в музичних рекомендаційних системах, які працюють на принципі фільтрації вмісту, об'єктами є музичні записи, які мають такі ознаки, як жанр, виконавець, рік створення, тип бек-вокалу, використані інструменти і так далі. Причому за наповнення всіх ознак в такому випадком відповідають музичні аналітики. Профіль користувача також може мати багато ознак, але вони заповнюються зазвичай особисто користувачем, наприклад в ході відповідей на запитання анкети про музичні переваги. Також рекомендаційна система може заповнювати профіль користувача, аналізуючи його минулі оцінки. Після того як профілі створені, рекомендаційна система,

З одного боку, використовують даний підхід рекомендаційні системи забезпечують точні рекомендації, тому що вони покладаються виключно на оцінки, дані об'єктів, і переваги саме конкретного користувача. Іншими словами, рекомендації користувачеві не залежать від оцінок інших користувачів. Також, системи даного типу здатні рекомендувати об'єкти, які раніше ніким не були

оцінені, інакше кажучи, такі системи не страждають від проблеми неоцінених об'єктів.

З іншого боку, розглядаються системи мають і суттєві недоліки. Головним з них є труднощі створення профілів об'єктів. Це обумовлено тим, що це вимагає величезних зусиль експертів і збір великої кількості інформації по кожному об'єкту. Іншим недоліком таких систем є те, що вони рекомендують об'єкти, схожі на ті, які раніше були високо оцінені користувачем, тобто вони не рекомендують щось кардинально нове.

2.1.2. Визначення користувацьких переваг через коллаборативну фільтрацію

Інший поширений підхід при побудові рекомендаційних систем – коллаборативна фільтрація. При цьому підході не потрібно створювати профілі користувачів і об'єктів, як в розглянутому вище підході. Цей підхід базується на зібраних даних про минулої активності і дії користувачів. Рекомендаційні системи на основі коллаборативной фільтрації можна розділити на два типи: системи, що використовують методи «сусідства», і використовують методи «латентних факторів».

Головна ідея рекомендаційних систем, що працюють на методах «сусідства», полягає в тому, що вони оцінюють взаємозв'язок між об'єктами, або між користувачами. Користувач-орієнтовані системи намагаються знайти користувачів зі схожими уподобаннями, тоді як об'єкт-орієнтовані системи оцінюють, чи сподобається користувачу новий об'єкт на основі його минулих оцінок «сусідніх» об'єктів, тобто об'єктів, близьких за оцінками іншими користувачами до даного.

Системи, що використовують методи «латентних факторів», намагаються виявити приховані залежності в оцінках об'єктів користувачами для того щоб розділити користувачів на групи. При цьому використовуються різні математичні методи класифікації і кластеризації. Після розбиття користувачів на групи, якщо кільком користувачам в одній групі сподобався новий об'єкт, то рекомендаційна система порекомендує цей об'єкт іншим користувачам в цій групі.

Підхід на основі коллаборативной фільтрації також має свої недоліки. Однією з найгостріших проблем є «проблема холодного старту». Це означає, що користувачеві необхідно спочатку оцінити досить велика кількість об'єктів, перш ніж рекомендаційна система зможе дати хороші рекомендації. Іншою важливою проблемою є розрідженість даних, що дуже сильно впливає на продуктивність таких рекомендаційних систем, так як це сильно знижує ймовірність знаходження користувачів (або об'єктів) зі схожими оцінками. Ця проблема виникає внаслідок того, що рекомендаційні системи зазвичай працюють з величезною кількістю об'єктів і користувачів. Наприклад, конкретний користувач просто фізично не може подивитися і оцінити більшість виходять в прокат кінофільмів.

2.2. Аналіз архітектури нейронних мереж для визначення користувацьких переваг

Архітектур нейронних мереж велика кількість, і з часом з'являються нові. Розглянемо найбільш часто зустрічаються базові архітектури.

2.2.1. Персептрон

Персептрон – це найпростіша модель нейромережі, що складається з одного нейрона. Нейрон може мати довільну кількість входів (на рисунку 2.1 зображено приклад виду персептрона з чотирма входами), а один з них зазвичай тотожно дорівнює 1. Цей одиничний вхід називають зміщенням, його використання іноді буває дуже зручним. Кожен вхід має свою власну вагу. При надходженні сигналу в нейрон, як і було описано вище, обчислюється зважена сума сигналів, потім до сигналу застосовується функція активації і сигнал передається на вихід.

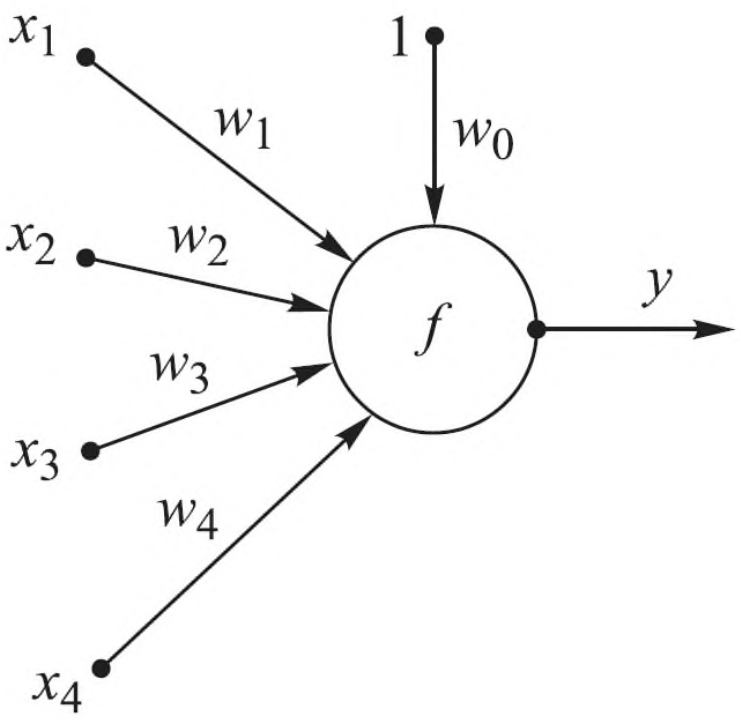

Рис. 2.1. Приклад архітектури персептрона

Така проста на перший погляд мережа, всього з одного нейрона, здатна, проте, вирішувати ряд завдань: виконувати найпростіший прогноз, регресію даних і т.п., а також моделювати поведінку нескладних функцій. Головне для ефективності роботи цієї мережі – лінійна роздільність даних.

2.2.2. Багатошаровий персептрон

Шаром називають об'єднання нейронів, на схемах, як правило, шари зображуються у вигляді одного вертикального ряду (в деяких випадках схему можуть повертати, і тоді ряд буде горизонтальним). Багатошаровий персептрон є узагальненням одношарового персептрона. Він складається з певної кількості вхідних вузлів, декількох прихованих шарів вичислительнихнейронов і вихідного шару (рис. 2.2).

Відмінні ознаки багатошарового персептрона:

– всі нейрони мають нелінійної функцією активації, яка є диференцуемою;

– мережа досягає високого ступеня зв'язності за допомогою синаптичних з'єднань$

– мережа має один або кілька прихованих шарів.

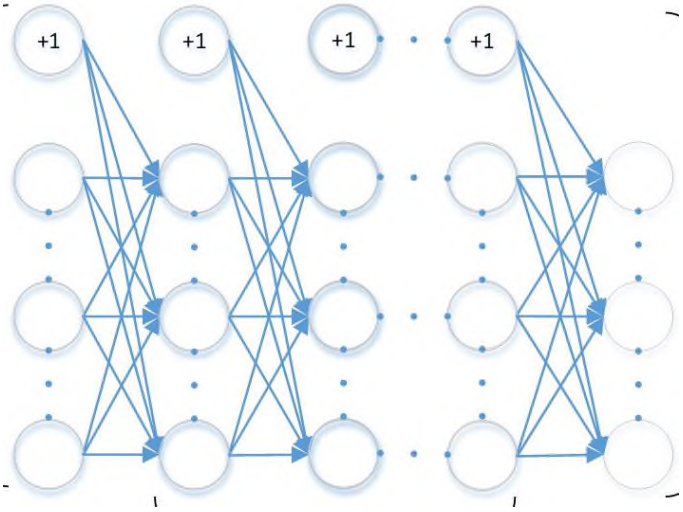

Рис. 2.2. Архітектура багатошарового персептрона.

Такі багатошарові мережі ще називають глибокими. Ця мережа потрібна для вирішення завдань, з якими не може впоратися персептрон. Зокрема, лінійно нероздільних завдань.

Насправді, як наслідок з теореми Колмогорова, саме багатошаровий персептрон є єдиною універсальною нейронною мережею, універсальним аппроксіматором, здатним вирішити будь-яке завдання. Можливо, не найефективнішим способом, який могли б забезпечити більш вузькопрофільні нейромережі.

2.2.3. Згорнута нейронна мережа

Згорнута нейронна мережа – мережа, яка обробляє дані, що передаються не цілком, а фрагментами. Дані послідовно обробляються, а після передаються далі по верствам. Згорткові нейронні мережі складаються з декількох типів шарів: сверточних шар, субдіскретізірующій шар, шар повно-мережі (коли кожен нейрон одного шару пов'язаний з кожним нейроном наступного – повна зв'язок). Шари згортки і підвибірки (субдискретизація) чергуються і їх набір може повторюватися кілька разів.

Назва архітектура мережі отримала через наявність операції згортки, суть якої в тому, що кожен фрагмент зображення множиться на матрицю (ядро) згортки поелементно, а результат підсумовується і записується в аналогічну позицію вихідного зображення. Необхідно це для переходу від конкретних

особливостей зображення до більш абстрактним деталям, і далі – кещё більш абстрактним, аж до виділення понять високого рівня (чи присутній що-небудь шукане на зображенні).

Згорткові нейронні мережі вирішують наступні завдання:

– класифікація. Приклад: нейронна мережа визначає, що або хто знаходиться на зображенні.

– детекція. Приклад: нейронна мережа визначає, що / хто і де знаходиться на зображенні.

– сегментація. Приклад: нейронна мережа може визначити кожен піксель зображення і зрозуміти, до чого він ставиться.

Описані мережі відносяться до мереж прямого поширення. Поряд з ними існують нейронні мережі, архітектури яких мають в своєму складі зв'язку, за якими сигнал поширюється в зворотну сторону.

2.2.4. Рекурентна нейронна мережа

Рекурентна нейронна мережа – мережа, з'єднання між нейронами якої утворюють орієнтований цикл. Тобто в мережі є зворотний зв'язок. При цьому інформація до нейронам може передаватися як з попередніх шарів, так і від самих себе з попередньої ітерації (затримка). Приклад схеми першої рекуррентной нейронної мережі (НС Хопфилда) представлений на рисунку 2.3.

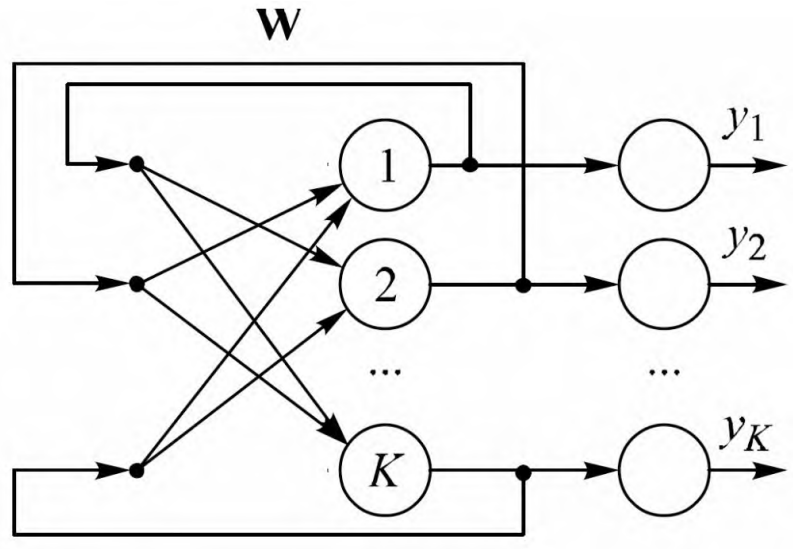

Рис. 2.3. Архітектура рекуррентной нейронної мережі.

Характеристики мережі:

– кожне з'єднання має свою вагу, який також є пріоритетом;

– вузли поділяються на два типи: вступні (зліва) і приховані (1, 2, ... *K*);

– інформація, що знаходиться в нейронної мережі, може передаватися як по прямій, шар за шаром, так і між нейронами.

Такі мережі ще називають «пам'яттю» (в разі мережі Хопфілда – автоасоціативною; бувають також гетероасоціативної (мережа Коско – розвиток мережі Хопфілда) та інші). Чому? Якщо подати даної мережі на вхід якісь «зразки» – послідовності кодів (наприклад, 1000001, 0111110 та 0110110) – і навчити її на запам'ятовування цих зразків, налаштувавши ваги синапсів мережі певним чином за допомогою правила Хебба, то потім, в процесі функціонування, мережу зможе «дізнаватися» після успішної реєстрації образи і видавати їх на вихід, в тому числі виправляючи спотворені подані на вхід образи. Наприклад, якщо після навчання такої мережі я подам на вхід 1001001, то мережу дізнається і виправить запомненний зразок, видавши на виході 1000001. Ця нейронна мережа не толерантна до поворотів і зрушень образів, і все ж для першої нейронної мережі свого класу мережу вельми цікава.

Таким чином, особливість рекуррентной нейронної мережі полягає в тому, що вона має «області уваги». Дана область дозволяє задавати фрагменти переданих даних, яким потрібно посилена обробка.

Інформація в рекурентних мережах з часом втрачається зі швидкістю, яка залежить від активаційних функцій. Дані мережі на сьогоднішній день знайшли своє застосування в розпізнаванні і обробці текстових даних. Про це мова піде далі.

2.2.5. Самоорганізована нейронна мережа

Приклад самоорганізується нейронної мережі – мережа Кохонена. У процесі навчання здійснюється адаптація мережі до поставленого завдання. У представленій мережі (рис. 2.5) сигнал йде від входу до виходу в прямому напрямку.

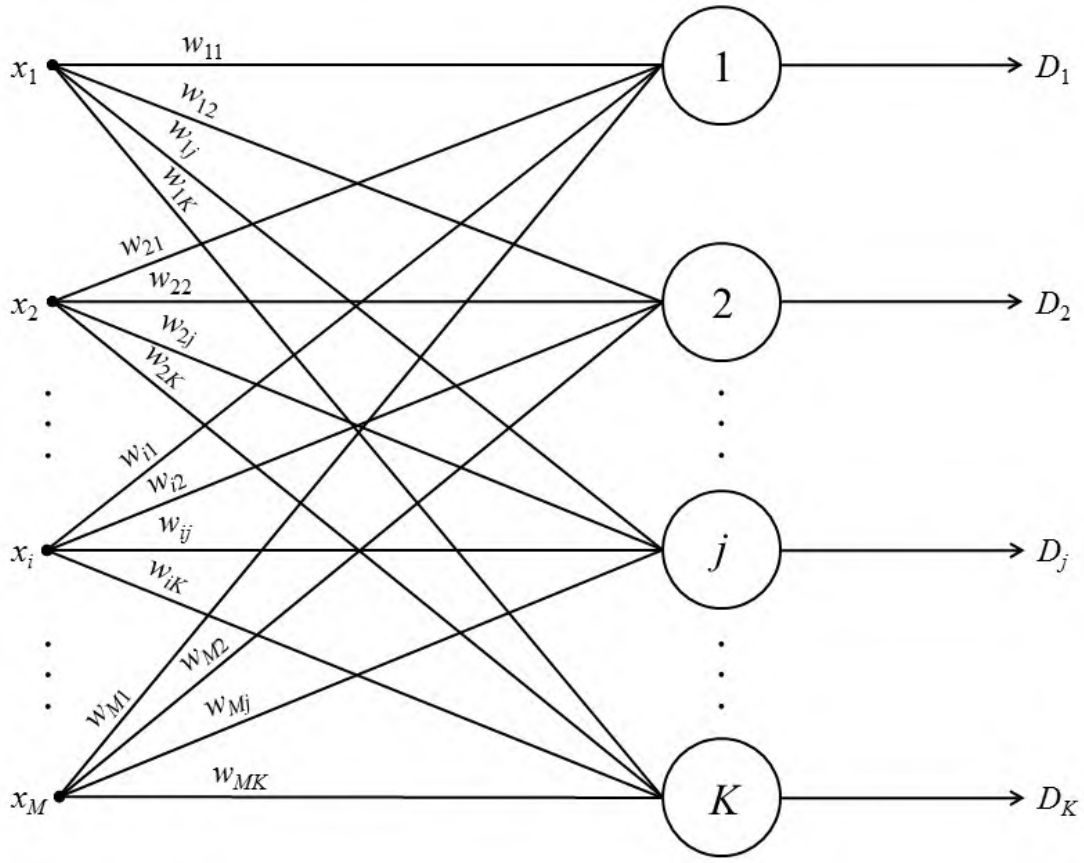

Рис. 2.5. Архітектура самоорганізується нейронної мережі

Структура мережі має один шар нейронів, які не мають коефіцієнтів зміщення (тотожне одиничних входів). Процес навчання мережі відбувається за допомогою методу послідовних наближень. Нейронна мережа підлаштовується під закономірності вхідних даних, а не під краще значення на виході. В результаті навчання мережу знаходить околиця, в якій знаходиться кращий нейрон. Мережа функціонує за принципом «переможець отримує все»: цей найкращий нейрон в результаті на виході матиме 1, а решта нейрони, сигнал на яких вийшов менше, 0. Визуализировать це можна так: уявіть, що правий ряд нейронів, вихідний, нарис. 7 – це лампочки. Тоді після подачі на вхід мережі даних після їх обробки на виході «запалиться» тільки одна лампочка, яка вказує, куди відноситься поданий об'єкт. Саме тому такі мережі часто використовуються в задачах кластеризації та класифікації.

2.2.6. Алгоритми навчання нейронних мереж

Щоб отримати рішення поставленого завдання з використанням нейронної мережі, спочатку потрібно мережу навчити. Процес навчання мережі полягає в налагодженні вагових коефіцієнтів зв'язків між нейронами. Алгоритми навчання для нейронної мережі бувають сучітелем і без.

З учителем: надання нейронної мережі деякої вибірки навчальних прикладів. Зразок подається на вхід, після відбувається обробка всередині нейронної мережі і розраховується вихідний сигнал, порівнюється з відповідним значенням цільового вектора (відомим «правильною відповіддю» для кожного з навчальних прикладів). Якщо відповідь мережі не збігається з необхідним, проводиться корекція ваг мережі, безпосередньо залежить від того, наскільки відрізняється відповідь мережі від правильного (помилка). Правило цієї корекції називається правилом Відроу-Хоффа і є прямою пропорційністю корекції кожного ваги і розміру помилки, похідною функції активації і вхідного сигналу нейрона. Саме після винаходу і доробок алгоритму поширення цієї помилки на всі нейрони прихованих шарів глибоких нейронних мереж ця область штучного інтелекту повернула до себе інтерес.

Без вчителя: алгоритм готує ваги мережі таким чином, щоб можна було отримати узгоджені вихідні вектори, тобто надання досить близьких векторів буде давати схожі виходи. Одним з таких алгоритмів навчання є правило Хебба, за яким настроюється перед роботою матриця ваг, наприклад, рекуррентной мережі Хопфілда.

Розглянемо тепер, які архітектури мереж є найбільш затребуваними зараз, які особливості використання нейромереж на прикладах відомих великих проектів.

2.2.7. Рекурентні мережі на прикладі *Google*

рекуррентная нейронна мережа (*RNN*) – мережа, яка володіє короткочасною «пам'яттю», з-за чого може швидко проводити аналіз

послідовностей різної довжини. *RNN* розбиває потоки даних на частини і проводить оцінку зв'язків між ними.

Мережа довготривалої короткостроковій пам'яті (*Long shortterm memory – LSTM*) – мережа, яка з'явилася в результаті розвитку *RNN*-мереж. Це цікава модифікація *RNN*-мереж, що дозволяє мережі не просто «тримати контекст», а й вміє «бачити» довготривалі залежності. Тому *LSTM* підходить для прогнозування різних змін за допомогою екстраполяції (виявлення тенденції на основі даних), а також в будь-яких завданнях, де важливим є вміння «тримати контекст», особливо добре підвладне для даної нейромережі (рис. 2.6).

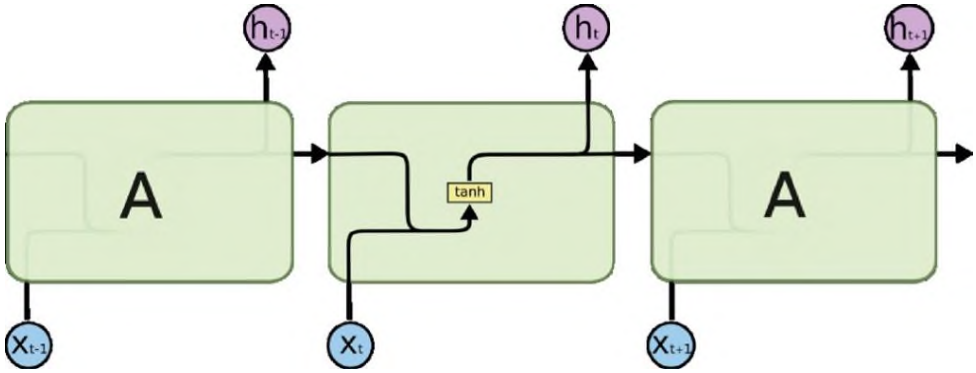

Рис. 2.6. Модулі рекуррентной нейронної мережі.

Форма рекурентних нейронних мереж є кілька повторюваних модулів. У стандартному вигляді ці модулі мають просту структуру. На малюнку 9 зображена мережа, яка має в одному з шарів гіперболічний тангенс як функції активації.

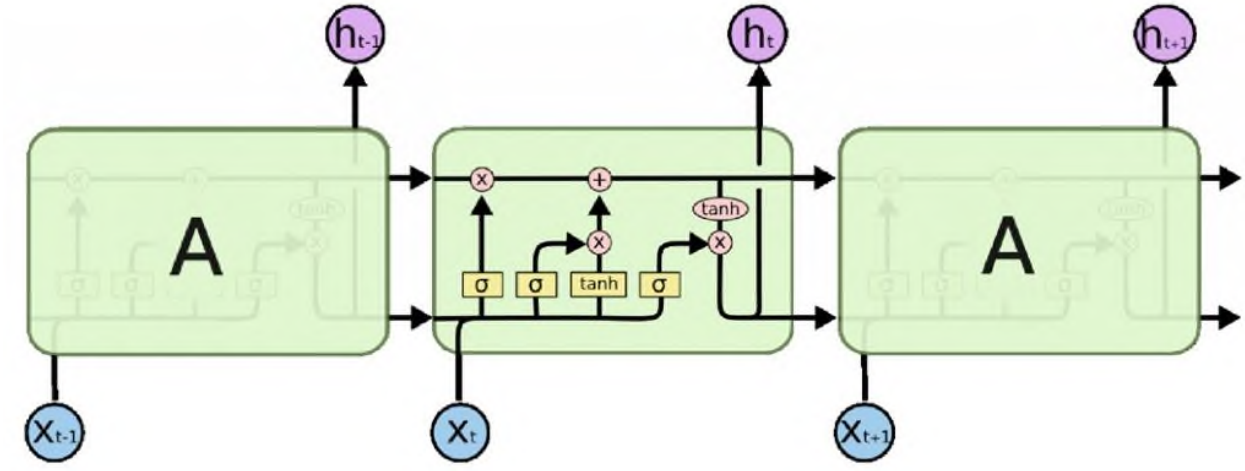

Рис. 2.7. Модуль *LSTM*-мережі.

LSTM також має цепочечную структуру. Відмінність полягає в тому, що мережа має чотири шари, а не один. Головною відмінністю *LSTM*-мережі є її клітинне стан (або стан осередку), яке позначається горизонтальною лінією у верхній частині діаграми і по якій проходить інформація (рис. 2.8). Це саме стан осередку нагадує стрічку конвеєра: вона проходить безпосередньо через весь ланцюжок, беручи участь в деяких перетвореннях. Інформація може як «текти» по ній, не наражаючись змін, так і бути піддана перетворенням з боку нейросеті7. (Уявіть собі лінію контролю на конвеєрі: вироби просуваються на стрічці перед співробітником контролю, який лише спостерігає за ними, але якщо йому щось здасться важливим або підозрілим, він може втрутитися).

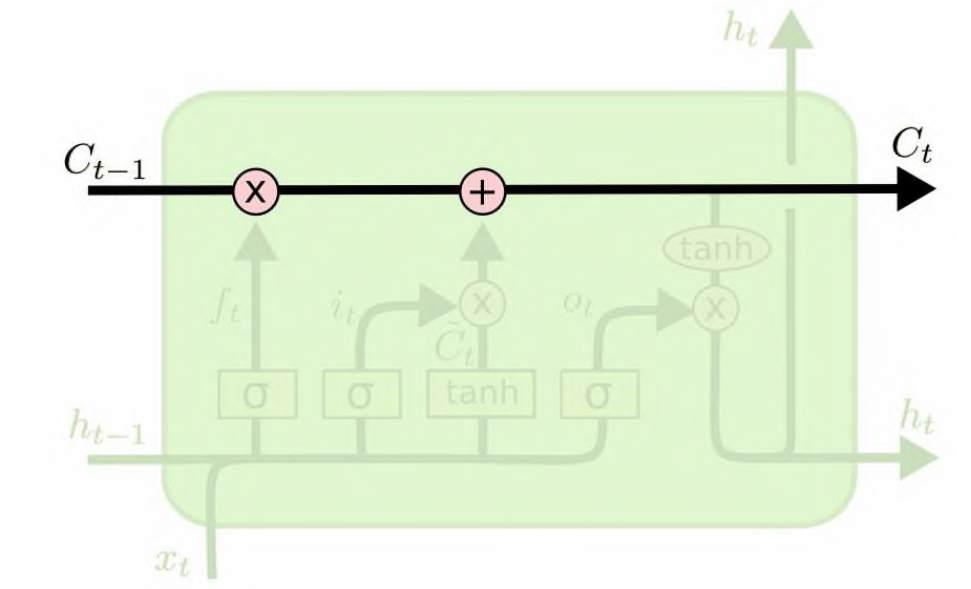

Рис. 2.8. Клітинний стан *LSTM*-модуля.

LSTM має здатність видаляти або додавати інформацію до клітинного стану. Дана здатність здійснюється за допомогою вентилів, які регулюють процеси взаємодії з інформацією. Вентиль – можливість вибірково пропускати інформацію. Він складається з сігмоідной шару нейронної мережі і операції поточечного множення (рис. 2.9).

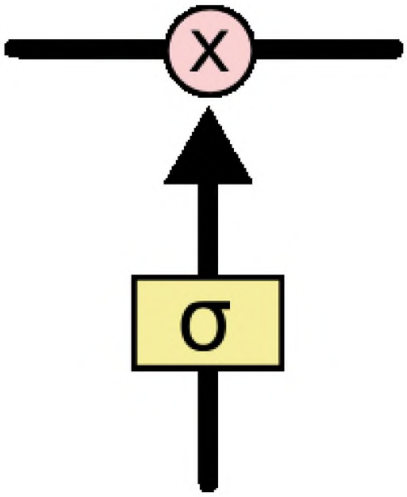

Рис. 2.9. Вентиль нейронної мережі

Сігмоідной шар (нейрони з сигмоидальной функцією активації) подає на вихід числа між нулем і одиницею, описуючи таким чином, наскільки кожен компонент повинен бути пропущений крізь вентиль. Нуль – «не пропускати зовсім», один – «пропустити все». Таких «фільтрів» в *LSTM*-мережі кілька. Для чого мережі потрібна ця здатність: вирішувати, що важливо, а що ні? Уявіть собі таке: мережа переводить пропозицію і в якийсь момент їй належить перекласти з англійської мови на російську, скажімо, займенник, прикметник або дієприкметник. Для цього мережі необхідно, як мінімум, пам'ятати рід і / або число попереднього іменника, до якого перекладається слово відноситься. Однак як тільки зустрінемо нове іменник, рід попереднього можна забути. Звичайно, це дуже спрощений приклад, тим не менш, він дозволяє зрозуміти труднощі прийняття рішення: яка інформація ще важлива, а яку вже можна забути. Інакше кажучи, мережі потрібно вирішити, яка інформація буде зберігатися в стані осередки, а потім – що на виході з неї буде виводити.

LSTM використовується в в *Google Translate*. *Google*-перекладач в даний час заснований на методах машинного навчання і використовує принцип роботи *LSTM*-мереж. Система робить обчислення, щоб зрозуміти значення слова або фрази, грунтуючись на попередніх значеннях в послідовності (контексті). Такий алгоритм допомагає системі розуміти контекст пропозиції і вірно підбирати переклад серед різних варіантів. Ще кілька років тому *Google Translate* працював на основі статистичного машинного перекладу – це різновид перекладу, де

переклад генерується на основі статистичних моделей (бувають: за словами, по фразам, по синтаксису і т.д.), а параметри цих моделей є результатами аналізу корпусів текстів двох обраних для перекладу мов. Ці статистичні моделі мали більшу швидкодію, однак їх ефективність нижче. Загальний приріст якості перекладу після впровадження системи на основі нейромереж в *Google*-перекладач склав, здавалося б, не дуже багато – близько 10%, проте для окремих мовних пар ефективність перекладу досягла 80-90%, впритул наблизившись до оцінок якості людського перекладу. «Платою» за таку якість перекладу є складність побудови, навчання і перенастроювання системи на основі нейромережі: вона займає тижні. Взагалі для сучасних глибоких нейронних мереж, що використовуються в таких великих проектах, в порядку речей навчання, що займає дні і тижні. «Платою» за таку якість перекладу є складність побудови, навчання і перенастроювання системи на основі нейромережі: вона займає тижні. Взагалі для сучасних глибоких нейронних мереж, що використовуються в таких великих проектах, в порядку речей навчання, що займає дні і тижні. «Платою» за таку якість перекладу є складність побудови, навчання і перенастроювання системи на основі нейромережі: вона займає тижні. Взагалі для сучасних глибоких нейронних мереж, що використовуються в таких великих проектах, в порядку речей навчання, що займає дні і тижні.

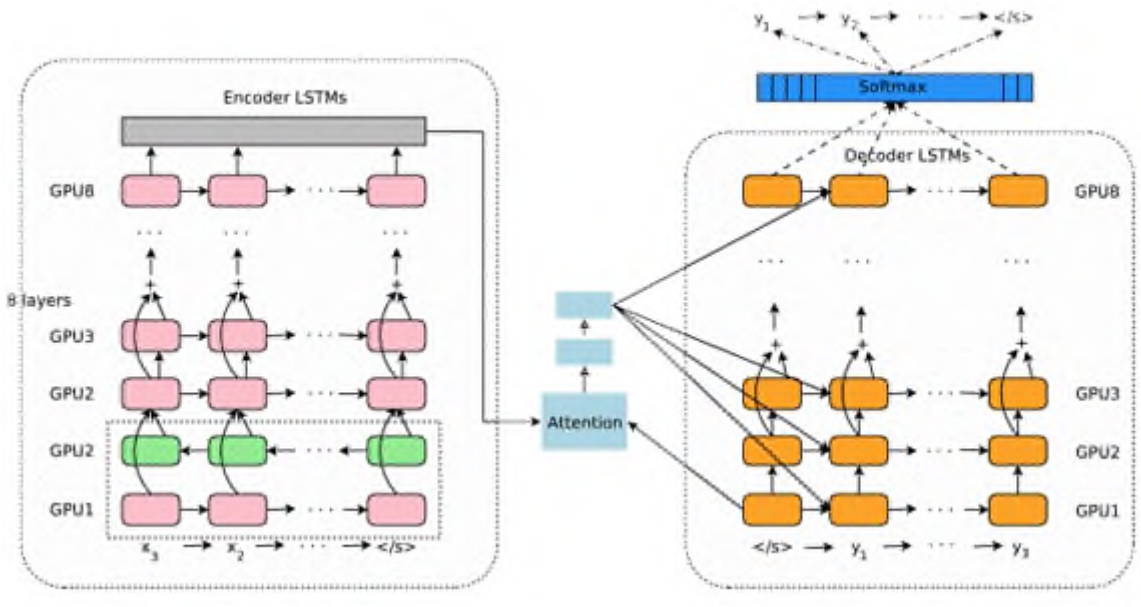

Рис. 2.10. Рекурентна двунаправленная нейронна мережа *Google*-перекладача

Розглянемо *LSTM*-мережу перекладача (*GNMT*) докладніше. Нейронна мережа перекладача має поділ на два потоки (рис. 2.10). Перший потік нейронної мережі (зліва) є аналізують і складається з восьми шарів. Даний потік допомагає розбивати фрази або пропозиції на кілька смислових частин, а після виробляє їх аналіз. Мережа дозволяє читати пропозиції в двох напрямках, що допомагає краще розуміти контекст. Також вона займається формуванням модулів уваги, які дозволяють другого потоку визначення пріоритетів окремо взятих смислових фрагментів.

Другий потік нейронної мережі (праворуч) є синтезують. Він дозволяє обчислювати самий підходящий варіант для перекладу, грунтуючись на контексті і модулях уваги (показані блакитним кольором).

В системі нейронної мережі найменшим елементом є фрагмент слова. Це дозволяє сфокусувати обчислювальну потужність на контексті пропозиції, що забезпечує високу швидкість і точність перекладача. Також аналіз фрагментів слова скорочує ризики неточного перекладу слів, які мають суфікси, префікси і закінчення.

2.3. Порівняння машинного навчання і глибинного навчання

Сучасні інтелектуальні системи машинного і глибинного навчання можуть коригувати параметри виконуваних ними операцій на основі аналізу безперервно одержуваних даних. У реальності ці області комп'ютерної науки мають ледь помітні відмінності.

Поняття машинного навчання (МН) застосовується до системи, яка може активно навчатися, а не просто пасивно отримувати інформацію для обробки. Така комп'ютерна система запрограмована реагувати на вхідні дані як людина. Для цього використовуються алгоритми аналізу даних для пошуку шаблонів і структур. Алгоритми МН розроблені таким чином, що їх продуктивність підвищується з часом завдяки отриманню все більшої кількості даних.

Коли людина розпізнає щось, це відбувається миттєво. Щоб імітувати цей процес, в алгоритмах машинного навчання використовуються нейронні мережі.

Як і в процесі навчання людини, комп'ютерна нейронна мережа класифікує дані (наприклад, великий набір фотографій) на підставі розпізнаних елементів в зображенні. При використанні «експертної» зворотного зв'язку від людини частка успішних спроб класифікації може рости з часом. Такий зворотній зв'язок допомагає системі навчатися і підтверджувати правильні рішення, відрізняючи їх від неправильних, що дозволяє досягти оптимальної ефективності і підвищити точність. Після цього алгоритм нейронної мережі змінює все майбутні рішення на підставі отриманого зворотного зв'язку. Цей процес імітує розпізнавання образів людиною шляхом **навчання** мережі для видачі необхідних вихідних даних.

Етапи машинного навчання:

1. Нейронна мережа може визначити наступне: а) будь-яка форма є повторюваним елементом; б) перелік можливих форм (рис. 2.11).

Рис. 2.11. Фігури для машинного навчання

2. Потім алгоритм застосовує отримані знання до даних, виконуючи пошук заданих елементів і розбиваючи їх на категорії (рис. 2.12).

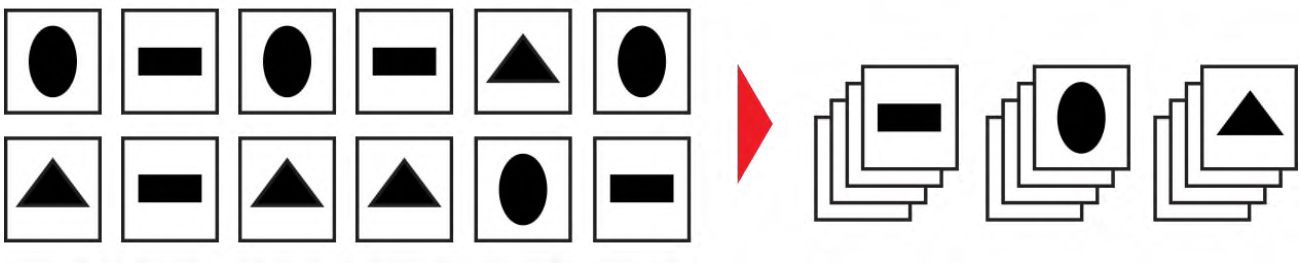

Рис. 2.12. Категорізація фігур для машинного навчання

3. З часом цей процес може вдосконалюватися завдяки зворотньому зв'язку від людини. Після цього алгоритм нейронної мережі змінює все майбутні рішення на підставі отриманого зворотного зв'язку. Це призводить до більш точному збору даних.

Наприклад, якщо надійшла зворотний зв'язок від людини «у кожної форми є кілька варіантів», алгоритм може впорядкувати результати зазначеним нижче чином.

Наприклад, компанія *Google* найняла професійних фотографів і документалістів, щоб вони виконували роль експертів при навчанні алгоритму на основі нейронної мережі, що використовується в інтелектуальній камері *Clips* цієї компанії. Зворотній зв'язок від людей дозволила зробити камеру інтуїтивно краще не тільки з точки зору технічних аспектів цифрової фотографії, але і завдяки передбачення більш абстрактних якостей зйомки пам'ятних моментів1.

Глибинне навчання (ГН) – це підмножина машинного навчання, яке глибше вирішує завдання. Воно працює аналогічно тому, як людський мозок розпізнає і згадує інформацію, і цей процес виконується без отримання даних від «експертів». Програмам ГН необхідний доступ до великих обсягів даних, на основі яких вони навчаються. В алгоритмах ГН використовуються глибокі нейронні мережі, які отримують доступ до величезних наборів інформації (наприклад, до всіх музичних файлів в *Spotify* або *Pandora*), вивчають і аналізують їх. В результаті глибокі нейронні мережі приймають рішення на підставі переваг конкретного користувача.

Етапи глибинного навчання:

1. Глибока нейронна мережа, яка використовується алгоритмами глибинного навчання, шукає дані в величезних наборах інформації, наданих їй для аналізу (рис. 2.13).

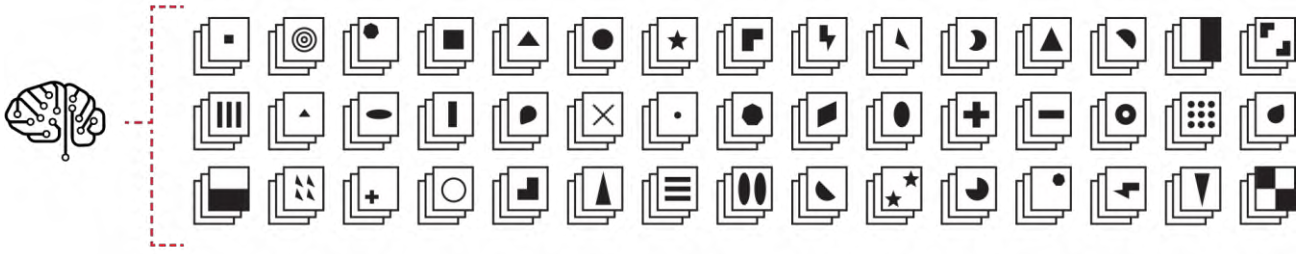

Рис. 2.13. Набори фігур для глибинного навчання

2. В процесі обробки цієї інформації глибока нейронна мережа розробляє нові параметри класифікації, наприклад: у форм можуть бути різні кольори або У форм може бути різна товщина.

3. Це призводить до того, що система починає видавати дуже точні результати, які з часом будуть ставати все точніше і точніше.

Основний фактор, який відрізняє ГН від МН, представлення даних. Наприклад, в описаному вище випадку МН для камери *Google Clips* для навчання системи були необхідні вхідні дані від професійних фотографів. В системі ГН для точного визначення властивостей експерти не потрібні. При оцінці вхідних даних, будь то зображення, новинна стаття або пісня, вони використовуються в необробленої формі, без міток і з мінімальними перетвореннями. Цей неконтрольований процес навчання іноді називають навчанням уявленням. В процесі навчання алгоритм ГН навчається на даних, щоб підвищити точність своїх висновків.

Нижче перераховані сучасні поширені приклади застосування ГН:

– автономне керування автомобілем: поєднання докладних даних (карт, супутникових знімків дорожнього руху, повідомлень про погоду, накопичених відомостей про переваги користувача), вхідних даних про навколишнє середовище, що надходять від датчиків в режимі реального часу (олень на дорозі або автомобіль, який виїхав зі своєї смуги), і обчислювальної потужності для прийняття рішень (зниження швидкості, поворот рульового колеса);

– медицина: дослідження в області ракових захворювань, наприклад навчання виявлення меланоми на фотографіях;

– Інтелектуальний будинок: інтелектуальні будинки, що використовують інтелектуальних особистих помічників і алгоритми розпізнавання голосу для розуміння унікальних голосових запитів користувачів та реакції на них;

– розваги: аналіз великої бібліотеки фільмів і даних телепередач (жанр, актори, режисери, огляди) і накопичення відомостей про смаки та вподобання користувачів;

– відвідування ресторанів: інтуїтивні рекомендації ресторанів на основі фізичного розташування користувачів, оглядів критиків, можливості бронювання столиків, заданих переваг і поведінки в минулому;

– інтернет. Таргетингової реклама, капча, складання новинних стрічок, блокування спаму, асоціативний пошук інформації;

– зв'язок. Маршрутизація пакетів, стиснення відеоінформації, збільшення швидкості кодування і декодування інформації.

Крім перерахованих областей, нейронні мережі застосовуються в робототехніці, комп'ютерних і настільних іграх, авіоніки, медицині, економіці, бізнесі та в різних політичних і соціальних технологіях, фінансово-економічному аналізі.

2.4. Описання підходу *Slope One*

Slope One- це один з найпростіших підходів до рекомендацій на основі коллаборативной фільтрації по схожості предметів, але в той же час точність рекомендацій алгоритму порівнянна з більш складними і ресурсоємними алгоритмами [6]. Він був розроблений Даніелем Лемайром і Ганною Маклахман в 2004 році і опублікований в 2005 році в статті [6].

Slope one буде використовуватися в даному дослідженні при проведенні експериментів для порівняння отриманих результатів від підходу на основі візерункових структур.

Метод *Slope One* працює з оцінками об'єктів, отриманих від користувачів. У нашому випадку дані представляють собою оцінки фільмів різними користувачами. Якщо – безліч фільмів (в загальному випадку, замість фільмів можуть бути будь-які інші об'єкти), – безліч користувачів, то оцінки оцінки користувача дані різних об'єктів зручно представити таблицею:

$$M = \{m_1, m_2, \dots, m_n\} U = \{u_1, u_2, \dots, u_k\}$$

Таблиця 2.1.

Оцінки фільмів користувачами

user \ *AudioMovie*	m_1	⋯	m_j	⋯	m_n
u_1	$r_{1,1}$	⋯	$r_{1,j}$	⋯	$r_{1,n}$
⋮	⋮		⋮		⋮
u_i	$r_{i,1}$	⋯	$r_{i,j}$	⋯	$r_{i,n}$
⋮	⋮		⋮		⋮
u_k	$r_{k,1}$	⋯	$r_{k,j}$	⋯	$r_{k,n}$

Кожен користувач подивився кілька фільмів і дав їм свою оцінку. оцінки$r_{i,j}$можуть бути цілими числами від до, або, якщо користувач не дивився фільм, то замість оцінки буде стояти, тобто відсутність оцінки.15 $*$

Тепер перейдемо безпосередньо до алгоритму:

1) на вхід алгоритм отримує безліч оцінок всіх користувачів (наприклад як в табл. 2.1.), Номер користувача, для якого будуть здійснюватися рекомендації. Також на вхід необхідно відправити ліву і праву межу (і) оцінки шуканих фільмів, тобто, якщо хочемо отримати в якості рекомендацій всі фільми з оцінками$left_borderright_border$від до , 45то ліва і права межа повинні бути 4і відповідно. І останнє: необхідно поставити5мінімальну і максимальну оцінки (і), допустимі в нашій задачі. Тут мається на увазі те, що якщо наприклад алгоритм спрогнозував для об'єкта оцінку, а максимально можлива оцінка, то повинні розуміти як.$min\_bordermax\_border6.5456.545$

В першу чергу знаходимо безліч всіх фільмів оцінених користувачем. $S(u_t)$

Для кожного неоціненої фільму користувачем виконуємо крок 4), тим самим вважаючи прогнозовану оцінку для фільму. Після цього переходимо до кроку 5).$m_j \in M \backslash S(u_t) u_t m_j$

Для кожного оціненого користувачем фільму, знаходимо – безліч користувачів, які переглянули фільми і. У разі якщо безліч непорожнє, тобто потужність множини, вважаємо відхилення: і додаємо до безлічі номер

$u_t m_i \in S(u_t) S_{j,i}(U \backslash \{u_t\}) m_i m_j S_{j,i}(U \backslash \{u_t\}) |S_{j,i}(U \backslash \{u_t\})| > 0 dev_{j,i} = \sum_{u_k \in S_{j,i}(U \backslash \{u_t\})} \frac{r_{k,j} - r_{k,i}}{|S_{j,i}(U \backslash \{u_t\})|} R_j i$

Після того, як підраховані всі існуючі відхилення, вважаємо прогнозовану оцінку. У разі, якщо безліч порожньо, спрогнозувати оцінку неможливо.

$$P(u_t)_j = \frac{1}{|R_j|} \sum_{i \in R_j} (dev_{j,i} + r_{t,i}) R_j = \{ i \mid m_i \in S(u_t), i \neq j, |S_{j,i}(U \backslash \{u_t\})| > 0 \} R_j$$

До цього кроку алгоритм підрахував всі можливі прогнозовані оцінки для фільмів з безлічі:

$$P(u_t) M \backslash S(u_t)$$

Алгоритм Рекомед всі фільми з оцінками, враховуючи мінімальну і максимальну допустимі оцінки:

$$left\_border \leq P(u_t)_j \leq right\_border$$

Тут необхідно зробити зауваження про те, що якщо потрібно отримати топ-*N* рекомендацій, то можна відсортувати всі отримані прогнозовані оцінки для об'єктів за спаданням, і в якості рекомендацій взяти перші *N* об'єктів.

2.4.1. Модельний приклад

Розглянемо роботу алгоритму *Slope One* на простому прикладі (табл. 2.2).

Таблиця 2.2.

Приклад даних для *Slope One*

user \ *AudioMovie*	m_1	m_2	m_3
u_1	5	3	2
u_2	3	4	*
u_3	*	2	5

Спробуємо передбачити оцінку фільму.$u_3 m_1$

Нехай

$left\_border = 4 right\_border = 5 min\_border = 1 max\_border = 5$

Знаходимо – безліч оцінених користувачем фільмів. $S(u_3) = \{m_2, m_3\}$

$$M \backslash S(u_3) = \{m_1\}$$

$$S_{1,2}(U \backslash \{u_3\}) = \{u_1, u_2\}$$

$$dev_{1,2} = \frac{(r_{1,1} - r_{1,2}) + (r_{2,1} - r_{2,2})}{|\{u_1, u_2\}|} = \frac{(5-3) + (3-4)}{2} = 0.5$$

$$S_{1,3}(U \backslash \{u_3\}) = \{u_1\}$$

$$dev_{1,3} = \frac{(r_{1,1} - r_{1,3})}{|\{u_1\}|} = \frac{(5-2)}{1} = 3$$

$$R_1 = \{2, 3\}$$

$$P(u_3)_1 = \frac{1}{|R_j|}\left(dev_{1,2} + r_{3,2} + dev_{1,3} + r_{3,3}\right) = \frac{1}{2}(0.5 + 2 + 3 + 5) = 5.25$$

З урахуванням максимальної допустимої оцінки, алгоритм передбачає оцінку для фільму, і отже рекомендує подивитися його користувачеві.$5 m_1 u_3$

2.4.2. Підхід на основі візерункових структур (*RAPS*)

Підхід до формування рекомендацій на основі візерункових структур головним чином базується на теорії, представленої в розділі 1. Було прийнято рішення дати йому назву *RAPS* (*Recommender Algorithm based on Pattern Structures*).

Даний метод, як і розглянутий вище метод *Slope One*, працює з оцінками об'єктів, отриманих від користувачів. Аналогічно, якщо – безліч фільмів, – безліч користувачів, то для користувача оцінки об'єктів зручно представити таблицею (табл. 2.1.). оцінки можуть бути цілими числами від до, або, якщо користувач не дивився фільм, то замість оцінки буде стояти, тобто відсутність оцінки.

$$M = \{m_1, m_2, \dots, m_n\} U = \{u_1, u_2, \dots, u_k\} \left(r_{i,j}\right)_{\substack{1 \le i \le k \\ 1 \le j \le n}} r_{i,j}$$

Опишемо сам алгоритм:

1) На вхід алгоритм отримує безліч оцінок всіх користувачів, номер користувача, для якого будуть формуватися рекомендації. Також на вхід необхідно відправити ліву і праву межу (і) оцінки шуканих фільмів, тобто, якщо хочемо отримати в якості рекомендацій всі фільми з оцінками $tleft\_borderright\_border$ від 4 до 5, то ліва і права межа повинні бути 4. і відповідно. 5.

2) Визначаємо безліч фільмів, які сподобалися користувачеві, тобто ті фільми, яким користувач поставив оцінку в інтервалі:

$$M_t = \left\{m_{t_1}, \dots, m_{t_q}\right\} left\_border \text{ до.} right\_border$$

3) На цьому кроці для кожного фільму, знаходимо безліч користувачів, яким цей фільм також сподобався:

$$m_{t_i} \in M_t A_{t_i} = [left\_border, right\_border]^{\square}_{m_{t_i}} \text{ для } 1 \le i \le q$$

В результаті отримаємо набір множин користувачів: $A_{t_1}, \dots, A_{t_q}$

4) для кожного безлічі застосовуємо формулу 1.3. з тим щоб знайти опис (візерунок), яке в нашому випадку буде інтервальним вектором:

$$A_{t_i}, 1 \le i \le q d_{t_i} = A^{\square}_{t_i} = \langle [a_1^{t_i}, b_1^{t_i}], \dots, [a_n^{t_i}, b_n^{t_i}] \rangle \quad \text{для } 1 \le i \le q$$

Тут важливо відзначити, що якщо у якогось користувача з безлічі відсутня оцінка якогось фільму $u_x A_{t_i} m_y$, Тобто, то ця порожня оцінка не враховується. $r_{x,y} = *$

5) На останньому кроці обчислюємо вектор $R = (R_1, \dots, R_n) \in \mathbb{R}^n$

$$R_j = \left|\left\{ i \mid 1 \le i \le q, \left[a_j^{t_i}, b_j^{t_i}\right] \subseteq [left\_border, right\_border] \right\}\right|,$$

Тобто для кожного фільму вважаємо скільки разів його опису потрапили в заданий інтервал. Якщо, то алгоритм рекомендує фільм до перегляду.

$$m_j \left[a_j^{t_i}, b_j^{t_i}\right] [left\_border, right\_border] R_j > 0$$

Зауважимо також, що якщо необхідно отримати топ-N рекомендацій, то можна відсортувати вектор частоти рекомендацій по спадаючій і в якості рекомендацій взяти перші N об'єктів.

ЧАСОВА складність даного алгоритму:

$$O(|M|)O(|M| * |U|)O(|M| * |U|)$$

2.5. Поняття музичного аудіо-стрімінгу і аналіз існуючих рішень

2.5.1. Поняття музичного аудіо-стримінгу

Більшість додатків музичного аудіо-стримінгу в Інтернеті пропонують користувачеві досить великий набір композицій, з яких він може вибрати потрібну і прослухати цікавлять його, а потім почути в автоматичному режимі інші твори того ж виконавця і, найчастіше, з того ж альбому. Також, не варто забувати, що на таких ресурсах прослуховування часто відбувається в режимі онлайн без необхідності очікувати, поки композиція повністю завантажиться. Це стандартний функціонал, властивий для даного типу додатків.

Також, додатки музичного аудіо-стримінгу можна умовно розділити на дві категорії: аудіотеки і радіо. Щоб зрозуміти, що все-таки відрізняє радіо від аудіотеки, варто згадати звичайне радіо, яке слухаємо, налаштувавши приймач на правильну частоту, в автомобілі, вдома, на роботі без використання мережі Інтернет. В такому радіо, як правило, дійсно вибираємо тільки частоту, що визначає станцію, яка підходить за уподобаннями та віщає в певному форматі, періодично розбавляючи мова ведучого музичними вставками. Вплинути на цей формат слухач може тільки побічно, замовляючи прямо під час ефіру улюблені пісні, за допомогою СМС-повідомлень, або іншим способом, передбаченим на радіо. Також існують радіо, які не проводять ефіри з провідними, а заповнюють весь період мовлення музикою певного формату,

Найчастіше, в функціонал аудіотеки не входить нічого, крім пошуку музики, яку запитує користувач, що вельми обмежує список цікавлять користувача композицій і вельми слабо впливає на розширення даного списку, так як будь-яка ймовірність попадання в поле зору незнайомих композицій вкрай мала. Радіо же постійно пропонує прослуховування незнайомі композиції, часом дуже різні, але не виходять за рамки формату. Проблема такого підходу, мабуть, в тому, що композиції навіть одного формату можуть мати достатні відмінності, щоб розходитися з уподобаннями слухача більше, ніж потрібно.

Спираючись на виявлені вище особливості, можна визначити музичний аудіо-стрімінг, як відтворення музичних композицій для кінцевого користувача за допомогою послідовної завантаження композиції протягом прослуховування.

Далі, щоб остаточно завершити опис веб-додатків музичного аудіо-стримінгу з усіма їх особливостями, розглянемо конкретні приклади музичних сервісів, що відносяться до категорії радіо, так як даний тип додатків безпосередньо має справу з рекомендаційними системами, але, через досить великої кількості таких сервісів, візьмемо тільки найбільш відомі з них.

Перед тим як розглядати аналоги, найбільш відповідають поставленим критеріям, хочеться згадати кілька не менш відомих музичних сервісів, які є аудіотекою. Йдеться про такі веб-додатках як *Last.fm*, *Google Play Music*, *ITunes* і їм подібних. Всі вони досить часто зараховуються до інтернет-радіо, але, навіть маючи певний схожий функціонал, в першу чергу, є аудіотекою, що поширюють музику на тих чи інших умовах.

Першим сайтом для розбору обрано *Pandora.com* (Рис. 2.14). На момент входу, *Pandora* пропонує користувачеві ввести якийсь запит, який містить назву пісні, ім'я виконавця або жанр музики. Після введення даних підбирається композиція, схожа на запитувану, але необов'язково є такою, і відкривається доступ до плеєра, де користувач може регулювати гучність, ставити музику на паузу, відзначати сподобалися і не сподобалися мелодії, а також пропускати композиції, але не може прокрутити їх на певний момент.

Також, поряд з плеєром присутній список призначених для користувача станцій і рядок пошуку для створення нових. Таким чином, *Pandora* повністю підходить під дане вище визначення додатки музичного аудіо-стримінгу.

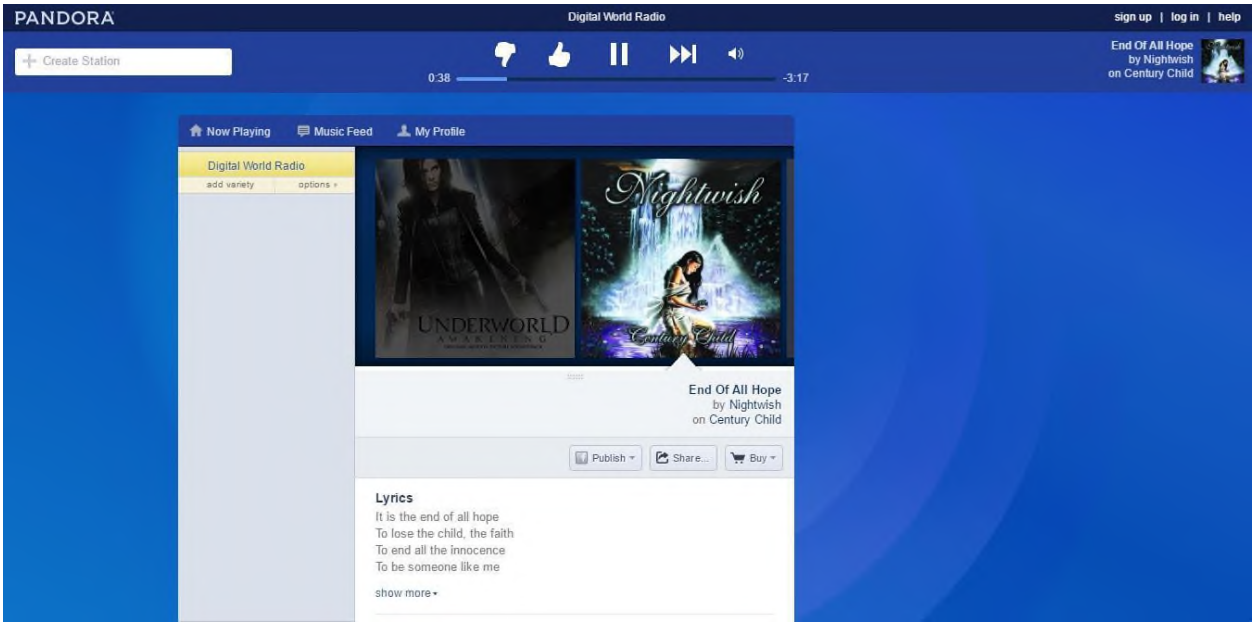

Рис. 2.14. Вікно сайту *Pandora Internet Radio*

З особливостей можна виділити *Music Genome Project* – рекомендаційну систему підбору композицій для кожного окремого користувача по вподобаним мелодіям. Особливим, в даному випадку, варто вважати не стільки сам принцип системи, скільки використовується алгоритм, який враховує близько 400 музичних параметрів, таких як гармонія, тональність і інші.

Звичайно, не обійшлося без мінусів. *Pandora* є американським сервісом і доступно тільки на території США, Австралії та Нової Зеландії, тому користувачі з інших країн, в тому числі України, повинні докласти певних зусиль і скористатися обхідними шляхами, щоб мати можливість використовувати послуги сайту. Також, програма надає досить великий набір композицій, але, виходячи з територіальних, і не тільки, обмежень, практично не включає в цей набір російських виконавців, що, хоч і не для всіх, буде ставитися до мінусів. В кінці, хотілося б згадати, що, хоча користуватися сайтом можна безкоштовно, на радіо діє обмеження на кількість пропусків композицій, яке відключаються оплатою преміум-режиму.

Другим в нашому розборі буде сервіс, який пішов від першого не так вже й далеко, а для кого-то є навіть більш відомим, ніж *Pandora*. З даними сайтів все трохи складніше. Справа в тому, що *Spotify* (Рис. 2.15) є аудіотекою, але включає також і непоганий функціонал радіо, як і *Pandora*, тому перше опустимо, а з другим будемо розбиратися докладніше.

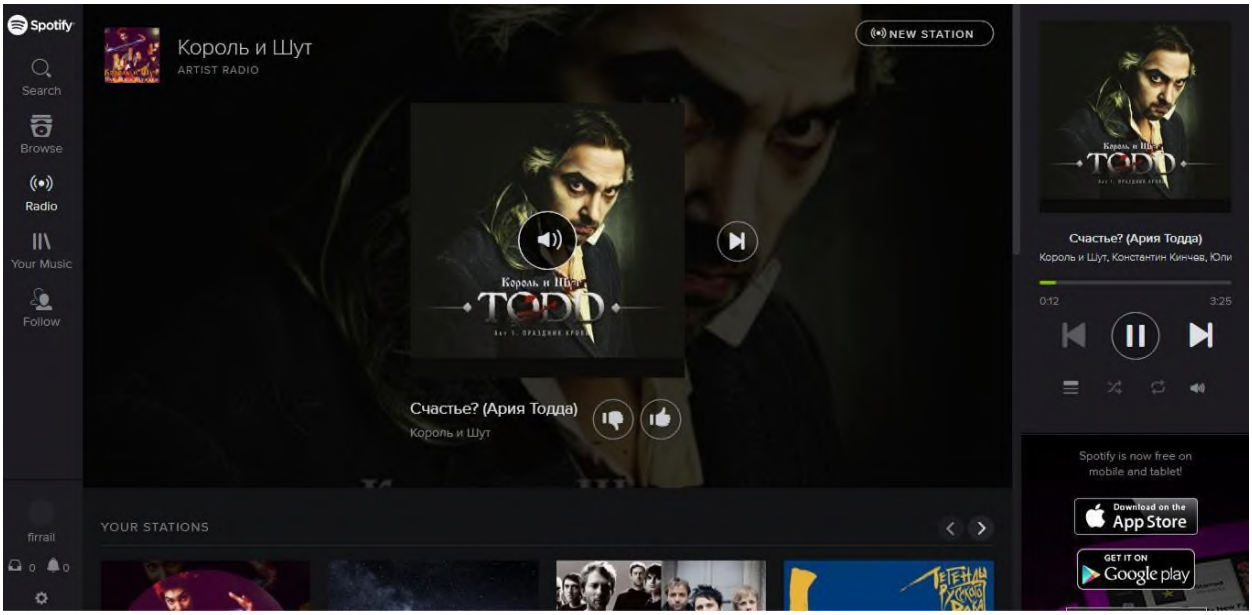

Рис. 2.15. *Spotify*, сторінка *Web Player*, вкладка *Radio*

Перше, що необхідно зробити перед використанням сервісу – зареєструватися, інакше скористатися його функціоналом не вдасться. Далі, користувачеві пропонують кілька станцій з рекомендованих виконавців, набір станцій за жанрами і можливість створити свою станцію за індивідуальним запитом. Після створення з'являється плеєр, де користувач все також може регулювати гучність, відзначати сподобалися композиції, ставити паузу, як і у всіх стандартних програвачах. На жаль, плеєр включає і інші стандартні функції, такі як прокрутка пісні, перехід до наступної або попередньої пісні, що не в'яжеться з концепцією радіо. Також, з такою концепцією не в'яжеться те, що в будь-який момент прослуховування користувачеві доступний програється плейлист, складений при створенні станції,

З особливостей можна відзначити настільний додаток для таких операційних систем, як *Windows, OS X, Linux*. Воно повністю повторює сторінку на сайті і має такий же функціонал, а користувачам з преміумом дозволяє

зберігати і прослуховувати музику в режимі *offline*. До речі, підписку на *Spotify* можна віднести до мінусів сервісу, так як, на відміну від *Pandora*, текстової рекламою і обмеженням пропусків справа не закінчується. Після реєстрації користувач отримує місяць пробного періоду без обмежень, а після його закінчення буде змушений прослуховувати між піснями рекламні вставки, що в століття розквіту капіталізму можна зрозуміти. Однак, крім реклами користувач без преміум-статусу буде слухати музику зі зниженим бітрейтом, а порізка якості композицій – це вже серйозний недолік. І, звичайно ж, не можна оминути того факту, що дивно, серед пропонованої музики крім зарубіжних є також і російські виконавці.

Таким чином, *Spotify* хоч і має функціонал радіо, але потрібен він для розширення можливостей що ставлять на перший план аудіотеки.

Останнім на черзі виступає, додаток *YouTube.Music* (рис. 2.16). Даний сервіс є продовженням програми Яндекс.Музика, яке є аудіотекою, але, на відміну від *Spotify*, виступає окремо, без жорсткої прив'язки функціоналу радіо до основного сервісу, а за концепцією більше передає дух *Pandora*.

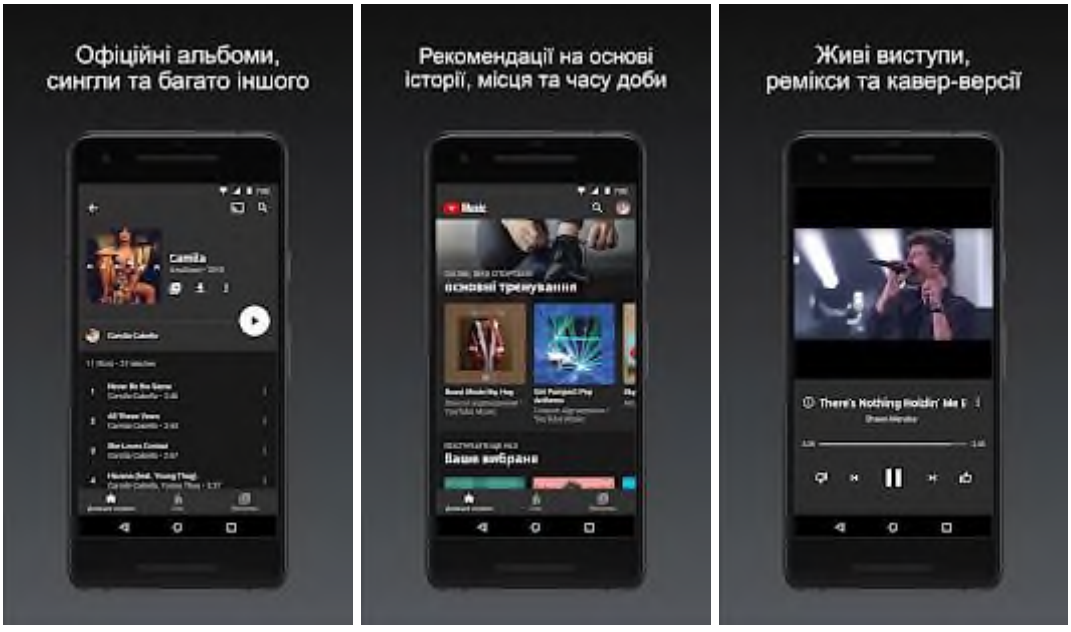

Рис. 2.16. Додаток *YouTube.Music*

Відвідуючи веб-додаток перший раз без авторизації, користувач зможе вибрати станцію з дуже і дуже великого списку за різними категоріями: жанр, заняття, настрій і навіть епоха. Після вибору станції з'являється плеєр з уже

звичними регулятором гучності, кнопкою паузи / відтворення і панеллю «подобається / не подобається». Також, як і на *Pandora*, тут немає функції прокрутки композиції або повернення до попередньої пісні, тільки пропуск всієї мелодії для переходу до наступної, яку, до речі, користувач бачить відразу, але тільки одну, а не весь плейлист, як це було на *Spotify*. Що ж стосується індивідуального підбору музики для кожного користувача – після реєстрації, прослуховуючи композиції різних станцій і відзначаючи сподобалися, починається поступове формування користувальницької станції, що відповідає його уподобанням і смакам.

Особливістю даного сервісу хотілося б виділити цікавий пульт управління на додаток до основного плеєра. Крім зміни підбору за такими критеріями як мову, де можна вибрати російську, іноземний або будь-який, і характеристика треків, популярних, улюблених, рідкісних або всіх підряд, в цьому пульті є два регулятора: енергійність, що змушує програму підбирати композиції спокійніше або бадьоріше, і настрій , що впливає на підбір більш сумною або веселої музики.

Незважаючи на настільки великий список станцій для вибору відповідного формату музики, на *YouTube.Music*.Радіо не передбачений пошук композицій за назвою і виконавцю, що може утруднити формування власної станції музики по перевагах, так як за весь час поневірянь по різних напрямках, можна так і не наштовхнутися на вподобану, але маловідому пісню улюбленого виконавця. Не стільки на мінус, скільки на невеликий недолік, але хотілося б звернути увагу на ті самі два регулятора в додатковому пульті управління, які роблять наступні композиції спокійніше або бадьоріше, веселіше або сумніше.

Вивчаючи цей пульт і змінюючи ці два регулятора, мною так і не було визначено за яким принципом ті чи інші композиції вважаються більш бадьорими або сумними, а також чому одна і та ж композиція може виступати і як сумна, але бадьора, і як сумна і спокійна. У першому випадку, ймовірно, питання досить суб'єктивний і вирішувати його потрібно опитуваннями великих груп людей, а в другому випадку, швидше за все, стоїть не менш складний, з точки зору, як особистого розуміння, так і реалізації, питання поєднання, в

деякому роді, протилежних характеристик. Виходячи з цього, назвемо цей пункт невеликим недоліком, але жодним чином не мінусом.

2.5.2. Визначення параметрів оцінювання музики

Властивості звуку залежать більшою мірою від частоти коливань, отже, будемо використовувати саме частотні показники для аналізу музичних композицій. Частота – це фізична величина, що дорівнює числу повторень або процесів на одиницю часу, вона вимірюється герцах. Відомо, що більшість людей мають можливість чути частотні коливання звуку від 20 *Гц* до 20 *КГц*. Звуки, що мають меншу частоту, називаються інфразвуком, а мають велику частоту – ультразвуком. При цьому інфразвукові коливання хоч і не можуть бути почуті, але можуть вловлювати осязательно.

Більшість використовуються в музиці звуків має частоти, що укладаються в межі від субконтроктави до п'ятої октави. Наприклад, звучання звичайного 88-клавішного фортепіано має діапазон від ноти «Ля» субконтроктави (приблизно 27 *Гц*) до ноти «До» п'ятої октави (4200 *Гц*). Також, не варто забувати, що весь спектр музичного звуку, як правило, включає не тільки чисті звуки основної частоти, але і гармонію звуків з частотами, які кратні основній частоті, а обертони музичних звуків розташовані у всьому діапазоні доступних для людського слуху частот.

Грунтуючись на цих даних, можна виділити кілька діапазонів частот, що уособлюють основні параметри кожної музичної композиції:

Низькі баси (10-80 *Гц*) – найнижчі ноти, які змушують все навколо резонувати. Якщо знехтувати цими звуками в музичній композиції, то вона буде здаватися ненасиченої, і звук втратить свою глибину.

Верхні баси (80-200 *Гц*) – верхні ноти басових інструментів, а також найнижчі для таких інструментів, як гітара. В даному діапазоні міститься велика частина енергії звукового ритму, через який з'являється бажання пританцьовувати під музику.

Нехтування цим діапазоном частот призведе до втрати відчуття сили звуку.

Низькі середні (200 – 500 *Гц*) – основний діапазон звучання гітари, він концентрує практично весь акомпанемент і ритм.

Середні середні (500 – 2500 *Гц*) – діапазон вокалу, а також соло деяких інструментів (гітара, скрипка, фортепіано). Без цих частот музичний твір стає вельми нудним і занудним.

Верхні середні (2,5 – 5 *КГц*) – містить більшість обертонів і гармонік, тут зосереджені самі верхні ноти фортепіано і ще декількох інструментів. Такі частоти дозволяють зробити звучання яскравим і іскристим, створюючи ефект присутності.

Низькі високі (5 – 10 *КГц*) – тут зустрічається сильне спотворення високих частот. Незважаючи на те, що людський слух здатний вловлювати частоти до 20 *КГц,* даний діапазон вважається межею сприйняття. Однак, якщо обмежитися цими частотами звук не буде достатньо хорошим.

Верхні високі (10 – 20 *КГц*) – остання октава, найтонші високі частоти. Без цього діапазону частот композиція може викликати деякий дискомфорт при прослуховуванні.

Кожна частота в спектрі має значення, що вимірюється в децибелах. За цим значенням можна судити про переважання тих чи інших частот в окремо взятій музичної композиції, з точки зору рекомендаційних систем, що використовують методи фільтрації вмісту, номера частот виступатимуть, як ключові слова в алгоритмі підбору, а значення сили звуку для кожної конкретної частоти – вагами ключових слів.

2.6. Спільна фільтрація за допомогою *FastAI*

Модель спільної фільтрації / система рекомендацій прагне передбачити рейтинг або переваги, які користувач надав би товару, враховуючи його старі рейтинги або переваги. Системи рекомендацій використовуються майже кожною великою компанією з метою підвищення якості своїх послуг.

Після завантаження набору даних з *Kaggle* потрібно завантажити модуль фільтрації колабсів *FastAI*, вказати шлях до набору даних і завантажити в *csv*, що містить рейтинги, а також *csv*, що містить інформацію про переваги.

Завантаживши дані, можемо створити *CollabDataBunch*, який є набором даних, спеціально створеним для проблем спільної фільтрації. передамо йому наші рейтингові дані, випадкове насіння, а також розмір нашого набору перевірки, який визначається аргументом *valid_pct*.

Можемо показати партію наших даних методом *show_batch* (рис. 2.17).

book_id	user_id	target
1470	36544	2.0
661	5461	4.0
596	9285	4.0
3952	17522	3.0
6628	10394	3.0

Рис. 2.17: Випадкова партія даних

Останнє, що потрібно зробити перед створенням та навчанням моделі, це отримання максимальних та мінімальних значень наших рейтингів. Потім передамо ці значення нашій моделі, щоб вона могла тоді стиснути кінцеві результати між цими двома значеннями.

FastAI пропонує два різні типи колабів. Проста модель під назвою *EmbeddingDotBias*, яка використовувалася майже для всіх систем рекомендацій кілька років тому. Він створює вкладення як для користувачів, так і для музичних композицій, а потім бере їх точковий добуток. Другий – це модель, заснована на нейромережі, яка використовує вбудовування та повністю зв’язані рівні.

Вбудовування – це відображення від дискретних об’єктів, таких як слова чи ідентифікатори музичних композицій та користувачів у нашому випадку, до вектора неперервних значень. Це може бути використано для пошуку схожості

між дискретними об'єктами, що не було б очевидним для моделі, якби вона не використовувала вбудовані шари.

Ці вектори вбудовування мають низькі розміри та оновлюються під час навчання мережі.

Обидві моделі можна створити за допомогою класу *collab_learner*. Зазвичай аргумент *use_nn* має значення *false,* і тому створюємо модель *EmbeddingDotBias*.

В якості подальших аргументів можемо передати учаснику колабу аргумент *n_factors*, який представляє розмір векторів вбудовування, а також аргумент *yrange*, який визначає діапазон значень рейтингу, які знайшли раніше.

Тепер можемо знайти швидкість навчання, навчити модель за допомогою методу *fit_one_cycle* і зберегти модель (рис. 2.18 та рис. 2.19).

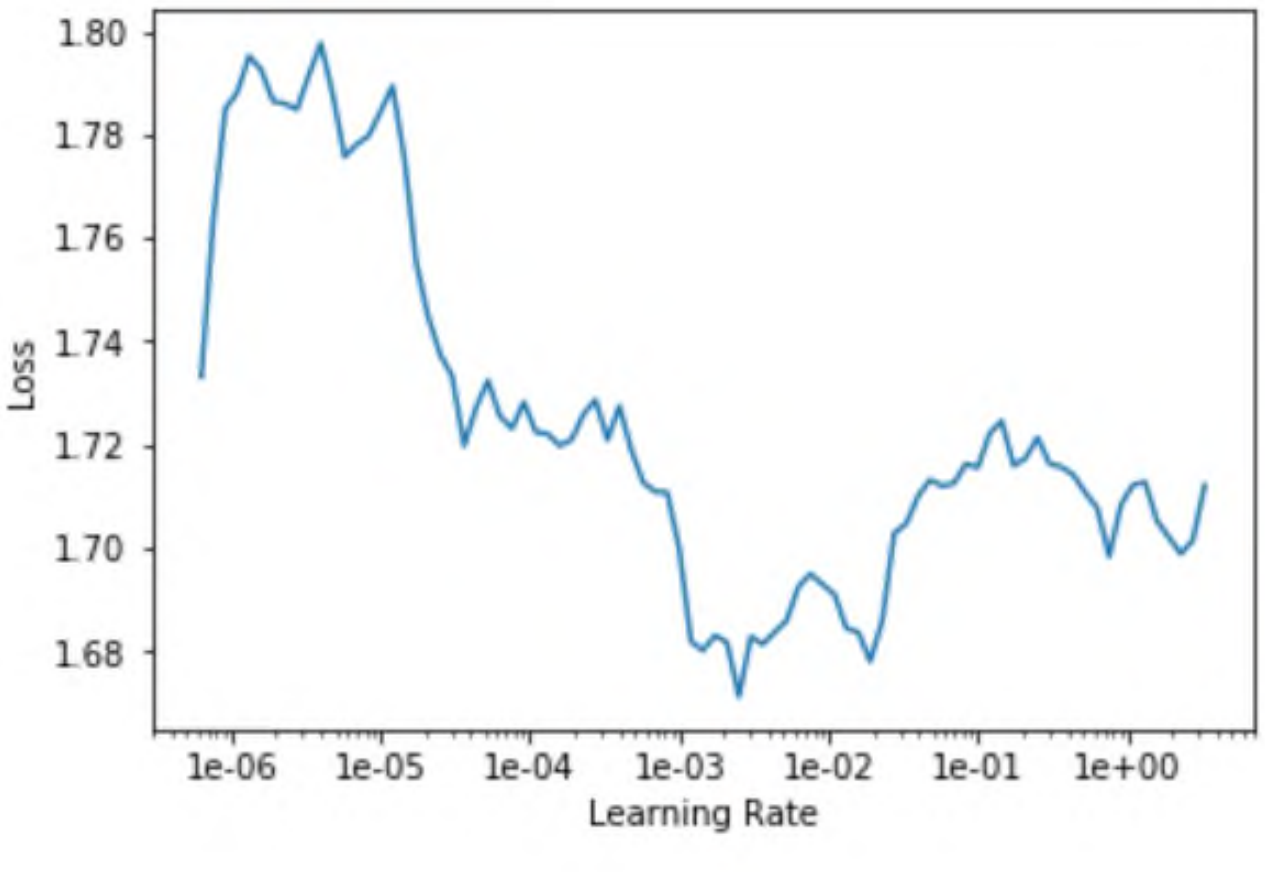

Рис. 2.18. Графік темпів навчання

Total time: 11:48

epoch	train_loss	valid_loss	time
1	1.574284	1.588789	02:21
2	1.315979	1.313377	02:21
3	1.133663	1.126915	02:22
4	1.059580	1.062018	02:22
5	1.045705	1.052467	02:21

Рис. 2.19 Результати навчання

Другий тип моделі спільної фільтрації, наданий *FastAI*, називається *EmbeddingNN*. Це надає можливість створювати вставки різного розміру та подавати їх у нейронну мережу. *FastAI* також надає можливість налаштувати кількість шарів та їх одиниць. Наступними кроками є пошук рівня навчання та навчання моделі (рис. 2.20 та рис. 2.21).

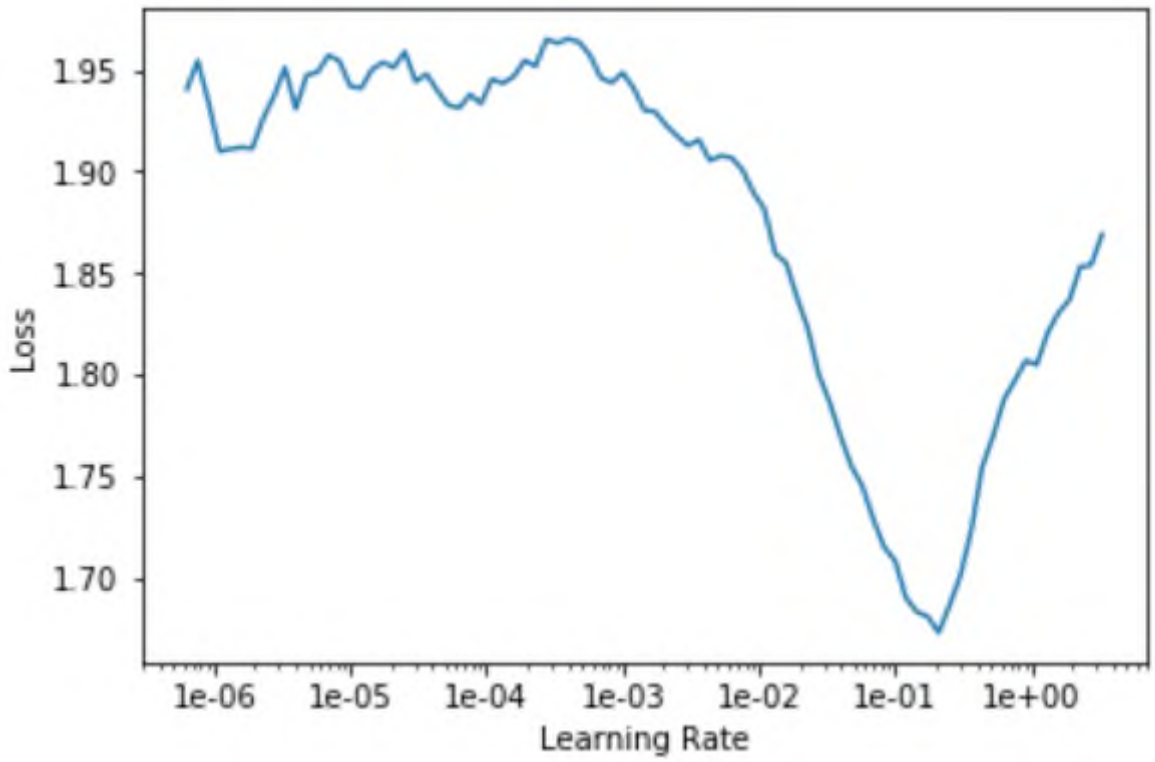

Рис. 2.20. Графік темпів навчання

Total time: 13:47

epoch	train_loss	valid_loss	time
1	0.763226	0.761729	02:45
2	0.747357	0.743163	02:46
3	0.702217	0.725356	02:45
4	0.675362	0.704841	02:45
5	0.571993	0.717858	02:45

Рис. 2.21. Результати навчання

Бачимо, що нейронна мережа працювала набагато краще, ніж наша точка-продукт. Оскільки вивчені вбудовування повинні якнайкраще відображати стиль і вид музичних композицій та користувачів, вони можуть містити цікаві функції, які можемо витягти та візуалізувати для того, щоб потрапити всередину.

З цією метою *FastAI* дозволяє легко отримати доступ до вкладених даних користувача та музичної композиції, а також до їх упереджень.

У цій статті виділимо упередження та ваги музичних композицій, щоб отримати уявлення про те, які музичні композиції слід класифікувати нижче або вище, використовуючи ухил вбудовування, а також про те, наскільки подібні деякі найпопулярніші музичні композиції за допомогою ваг вбудовування.

Для початку завантажимо нашу модель *EmbeddingDotBias* і отримаємо 1000 найпопулярніших музичних композицій за кількістю завантажень.

Тепер можемо виділити упередження для найкращих музичних композицій, а також середній рейтинг найкращих музичних композицій і роздрукувати їх. Завдяки цьому можемо отримати інформацію про музичні композиції, які, як правило, оцінюються як низькі чи високі, незалежно від того, який користувач їх оцінює.

2.7. Висновки до розділу

У цьому розділі були розглянуті основні підходи до рекомендацій: коллаборативна фільтрація і фільтрація вмісту. Потім було розглянуто метод рекомендацій *Slope One*, який в подальшому буде використовуватися як базовий при проведенні експериментів. Був представлений розроблений підхід до формування рекомендацій на основі візерункових структур (*RASP*).

Додатково було проаналізовано приклади оцінки вподобань користувачів для музичних систем.

РОЗДІЛ 3
ПРОЕКТУВАННЯ НЕЙРОННОЇ МЕРЕЖІ ВИЗНАЧЕННЯ КОРИСТУВАЦЬКИХ ПЕРЕВАГ

3.1. Етапи машинного експерименту

3.1.1. Дані для розрахунків

Для практичних випробувань розробленого методу рекомендацій на основі візерункових структур (*RAPS*), було прийнято рішення скористатися вільно поширюваними даними оцінок фільмів, які були зібрані з вересня 1997 року по квітень 1998 року за допомогою веб *AudioMovieLens* (*AudioMovielens.umn.edu*). Збір даних проходив в рамках проекту *The GroupLens Research Project* Університету Міннесоти.

В обраних даних представлено 100000 оцінок по 1682 фільмів від 943 різних користувачів. Причому кожен користувач в цих даних поставив оцінки не менше, ніж 20 фільмів.

Дані являють собою 100000 рядків виду:

user id | *item id* | *rating* | *timestamp*

Тобто в кожному рядку записано: номер користувача, номер фільму, оцінка, яку користувач поставив цього фільму, і час, коли це сталося.

3.1.2. Точність і повнота

Для порівнювання методів рекомендацій *Slope One* і *RAPS* спочатку було прийнято рішення скористатися стандартними заходами точності (*precision*) і повноти (*recall*):

$$precision = \frac{\left|\{relevant\ movies\} \cap \{retrieved\ movies\}\right|}{\left|\{retrieved\ movies\}\right|} \quad (3.1)$$

$$recall = \frac{\left|\{relevant\ movies\} \cap \{retrieved\ movies\}\right|}{\left|\{relevant\ movies\}\right|} \quad (3.2)$$

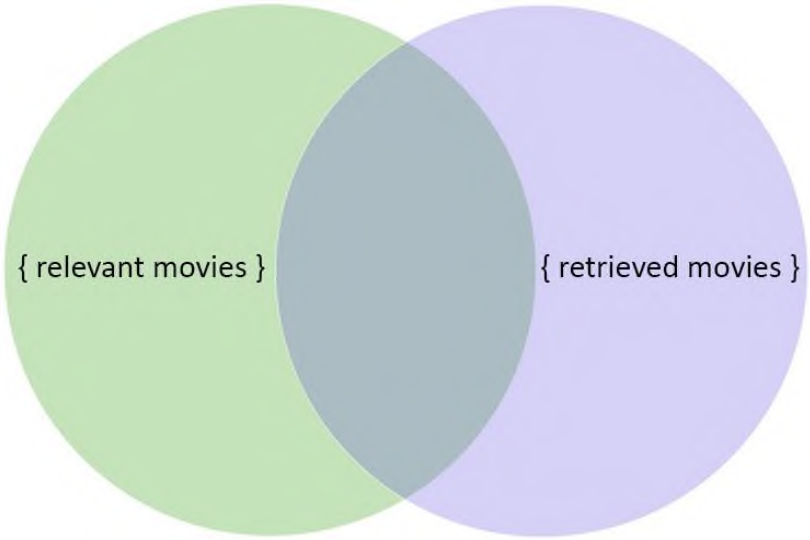

Рис. 3.1. Безлічі рекомендованих і релевантних музичних кліпів

Але вже в ході перших проведених експериментів стало видно, що точність методу *RAPS* виходить, в порівнянні зі *Slope One*, набагато меншою. Причина була в тому, що *RAPS* кожному користувачеві в середньому видавав на порядок більше рекомендацій, ніж *Slope One*, і більшість з цих рекомендованих музичних композицій користувач не дивився, і отже не оцінював.

Стало зрозуміло, що так оцінювати методи не зовсім коректно. Дійсно, якщо метод порекомендував музичну композицію, а користувач її ще не оцінив, то не можемо точно знати, сподобається він йому чи ні насправді.

Тоді для порівнювання методів рекомендацій *Slope One* і *RAPS* була розроблена модифікація точності (*precision*) і повноти (*recall*), які будемо називати скоригованими точністю і повнотою.

Для кожного конкретного користувача безліч прослуханих їм музичних композицій розбивається на два безлічі: безліч музичних композицій, на яких навчаються алгоритми (*train AudioMovies*), і безліч контрольних музичних композицій (*test AudioMovies*). Перше безліч складалося з 80% прослуханих музичних композицій, а друге відповідно з 20%. Причому фільми з першого безлічі були оцінені користувачем раніше, ніж фільми з другої множини. Для

такого розбиття всі оцінки користувача були впорядковані за часом проставлення і потім поділені.

Отже, було прийнято рішення скориговану точність і повноту розраховувати за формулами зазначеним нижче:

$$precision = \frac{\left|\{relevant\ movies\} \cap \{retrieved\ movies\} \cap \{test\ movies\}\right|}{\left|\{retrieved\ movies\} \cap \{test\ movies\}\right|} \quad (3.3)$$

$$recall = \frac{\left|\{relevant\ movies\} \cap \{retrieved\ movies\} \cap \{test\ movies\}\right|}{\left|\{relevant\ movies\} \cap \{test\ movies\}\right|} \quad (3.4)$$

Такі оцінки точності і повноти дозволили уникнути описану вище проблему невизначеності, так як не знаємо як саме користувач оцінить рекомендований фільм. У реальному житті б могли запитати користувача, чи вірна рекомендація, але в нашому випадку цього дозволити не можемо. Іншими словами, при оцінці точності і повноти даних методом вважаємо, що в даний конкретний момент, для заданого користувача, музичних композицій крім *train AudioMovies* і *test AudioMovies* просто не існує.

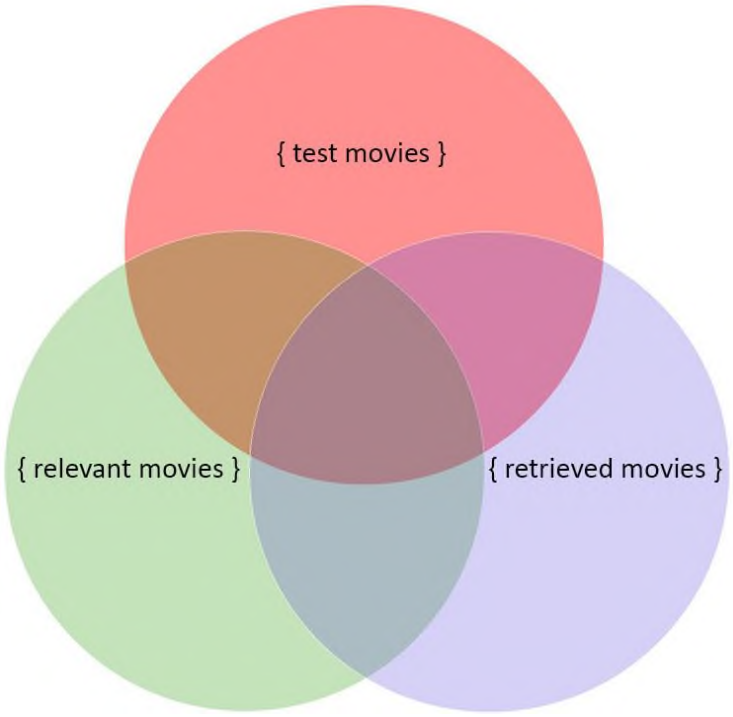

Рис. 3.2. Перетин безлічі рекомендованих, релевантних і протестованих музичних кліпів

3.2. Програмні засоби

Для виконання порівняльного аналізу методів *Slope One* і *RAPS* на мові програмування *MATLAB* було написано кілька програм:

1) *data_load.m*- ця маленька програма завантажує вихідних дані і з них створює дві розріджені (*sparse*) матриці: матрицю оцінок і матрицю часу. Ці матриці зберігаються як глобальні змінні і використовуються при запуску інших програм. Код програми з коментарями наведено в Додатку 1.

2) *main.m* – ця програма розбиває випадковим чином вихідна безліч користувачів на два: *train_users* (80%) і *test_users* (20%). Потім для кожного користувача з *test_users* виконуються наступні кроки: безліч прослуханих їм музичних композицій розбивається на безліч музичних композицій, на яких будуть навчатися алгоритми (*train AudioMovies*), і безліч контрольних музичних композицій (*test AudioMovies*) таким чином. Далі запускаються рекомендаційні алгоритми *RAPS* і *Slope One* з різними параметрами. І в кінці обчислюються точність і повнота так само, як описано вище за формулами (3.3) і (3.4). Код програми з коментарями наведено в Додатку А;

3) *raps.m*- ця функція є реалізацією алгоритму *RAPS*;

4) *slope_one.m*- ця функція є реалізацією алгоритму *Slope One*. Код функції з коментарями наведено в Додатку А.

5) *precision_and_recall.m* – ця функція обчислює скориговану точність і повноту за формулами за формулами (3.3) і (3.4). Код функції з коментарями наведено в Додатку А.

Необхідно зауважити, що перед усіма написаними програмами ставилася чисто дослідницька задача. Надалі вони будуть оптимізовані і вдосконалені.

Дана програма виконує наступні дії:

– розбиває безліч користувачів випадковим чином на дві множини *train_users* і *test_users* (80% і 20% відповідно);

– потім послідовно запускає алгоритм *RAPS* з параметрами хороших музичних композицій,,. Зберігає отримані вектори рекомендацій *number_of_recommendations* для подальшого підрахунку скоригованої точності і

повноти, а також зберігає витрачений час на кожен запуск. Тут потрібно зауважити, що алгоритм запускається для кожного користувача з тестової вибірки: 80% перших оцінених музичних композицій (використовуємо *timestamp*, в якому зберігатися час проставлення оцінки) йдуть на навчання методу, 20% останніх оцінених потім будемо використовувати для оцінювання результатів. [5,5][4,5][3,5];

– запускаємо алгоритм *Slope One* також для всіх користувачів з тестової вибірки. Аналогічно, 80% музичних композицій кожного користувача для навчання. Параметри хороших музичних композицій для даного алгоритму тут не важливі, так як він повертає вектор прогнозованих оцінок для кожного фільму *ratings* (який зберігаємо для кожного користувача). Рекомендовані фільми з цих оцінок обчислимо при виконанні наступного кроку;

– розраховуємо середню точність, повноту і час для алгоритмів *RAPS* і *Slope One* для параметрів хороших музичних композицій,,. [5,5][4,5][3,5];

– Виводимо отримані результати.

3.3. Експерименти і оцінка результатів

Як вже говорилося вище, для порівняння алгоритмів *RAPS* і *Slope One* скористалися формулами точності (3.3) і повноти (3.4).

Всі користувачі були випадковим чином поділені на дві множини: на 80% користувачів відбувалося навчання алгоритмів, а на 20% -тестування. Причому, для кожного користувача з контрольної вибірки безліч прослуханих їм музичних композицій також поділялося на два: 80% раніше прослуханих музичних композицій – для навчання алгоритмів, і 20% останніх прослуханих – для тестування.

Було проведено три серії тестів:

– Коли гарною оцінкою вважалася тільки оцінка (інтервал).5[5,5]

– Оцінка вважалася хорошою, якщо вона лежала в інтервалі.[4,5]

– Гарною оцінкою музичної композиції для рекомендації вважалася будь-яка оцінка в інтервалі.[3,5]

Всі тести проводилися на комп'ютері під керуванням *OS X* 10.9.3 з процесором *Intel Core i*7 2.3 *ГГц* і 8 *ГБ* оперативної пам'яті. Версія *MATLAB* – *R*2013*a*.

Середні значення результатів вийшли такими (табл. 3.1).

Таблиця 3.1

Результати експериментів

	Середній час, сек	Середня точність,%	Середня повнота,%
RAPS [5,5]	3,62	19,42	50,52
Slope One [5,5]	18,37	1,57	23,41
RAPS [4,5]	18,23	55,62	63,33
Slope One [4,5]	18,90	53,99	30,39
RAPS [3,5]	32,98	80,11	83,65
Slope One [3,5]	18,51	83,81	81,88

В якості критеріїв порівняння методів використовувалися: середній час роботи алгоритму в секундах, середня точність і середня повнота.

З таблиці з результатами чітко видно, що алгоритм *RAPS* сильно випереджає *Slope One* за всіма критеріями оцінювання для початкових параметрів [5,5].

Для параметрів обидва підходи мають схоже середній час роботи і точність, але *Slope One* видає в два рази менше повні рекомендації (параметр [3,5]).

Для параметрів алгоритми мають схожі середні значення точності і повноти, але *RAPS* в середньому виконує роботу в півтора рази повільніше (параметр [3,5]).

З усього цього можемо зробити висновок, що розроблений нами метод рекомендацій на основі візерункових структур має право на існування. Метод *Slope One* вже давно застосовується в реально працюючих рекомендаційних система [6], так що цілком можемо рекомендувати розробникам використовувати розроблений нами метод рекомендацій на основі візерункових структур.

Як метод побудови рекомендаційної системи був обраний *Grouplens* алгоритм. Даний алгоритм складається з двох етапів. На першому етапі відбувається визначення схожості користувача, для якого необхідно передбачити оцінки, які він може виставити, з іншими користувачами. Для цього використовується коефіцієнт кореляції Пірсона.

Під час другого етапу відбувається обчислення передбачуваного рейтингу альбомів, які ще не оцінив.

Алгоритм рекомендаційної системи починається з перевірки авторизації користувача. Якщо він не авторизований, то йому виводиться повідомлення про неможливість отримання рекомендацій і пропонується пройти її. Далі формуються запити до бази даних на отримання інформації про користувачів, альбомах і оцінках. Отримана інформація зберігається у вигляді одновимірних або двовимірних масивів. У масиві з оцінками поточного користувача індекс являє собою ідентифікатор альбому, а значення – оцінку, виставлену цього альбому. Аналогічним чином влаштований масив для оцінок інших користувачів, але в цьому випадку він буде двовимірним. У ньому кожен рядок відповідає якомусь користувачеві.

Після отримання з бази необхідних даних, здійснюється перевірка оцінок поточного користувача. Якщо їм не було оцінено жодного альбому, то система виводить повідомлення про неможливість отримання рекомендацій і пропонує виставити кілька оцінок. Якщо були оцінені всі альбоми, то виводиться повідомлення про неможливість отримання рекомендацій з проханням повторити спробу пізніше.

На наступному етапі визначається коефіцієнт схожості поточного користувача з усіма іншими. Для кожного користувача знаходяться альбоми, які були оцінені обома, після чого формула розбивається на три частини, і обчислюється кожна частина. Якщо хоча б одна частина, розташована в знаменнику, дорівнює нулю, то весь коефіцієнт кореляції між цими користувачами, прирівнюється нулю. Інакше всі частини підставляються в дану формулу і відбувається її обчислення.

Після того, як були визначені коефіцієнти подібності користувачів, виконується їх перевірка. Якщо всі коефіцієнти виявилися нульовими, то система виводить повідомлення про неможливість отримання рекомендацій і пропонує повторити спробу пізніше.

Далі для кожного альбому, який не був оцінений поточним користувачем, обчислюється прогнозований рейтинг. Якщо є користувачі, які виставили оцінку альбому, то формула (1.2) розбивається на 2 частини і обчислюється кожна з них. Якщо будь-яка з частин дорівнює 0, то прогнозований рейтинг прирівнюється середньому арифметичному оцінок поточного користувача. Інакше частини підставляються в формулу і обчислюється рейтинг.

Після всіх обчислень отримані прогнози заносяться в спеціальну таблицю, відправляється запит на отримання альбомів, відсортованих в порядку убування за рейтингом, відбувається отримання його результатів і оновлення сторінки. В результаті користувачеві надаються рекомендовані йому альбоми.

3.4. Використання розробленоъ нейронної мережі на сайті

Головна сторінка містить шапку сайту, слайдер зображень, панель пошуку по фільтрам і пошуку рекомендацій і каталог альбомів (рис. 3.3).

Шапка сайту, яка також міститься і на інших сторінках сайту, складається з логотипу і іконок «Увійти» і «Реєстрація». При натисканні на логотип можна перейти на головну сторінку, перебуваючи при цьому на будь-який інший. Натиснувши на іконку «Увійти» або «Реєстрація», користувач переходить на сторінку авторизації або реєстрації відповідно.

Слайдер зображень являє собою слайд-шоу з фотографій музичних груп і виконавців.

Панель пошуку по фільтрам дозволяє користувачеві знайти альбоми певних жанрів і років випуску, а також новинки, необхідно лише виставити необхідні параметри і натиснути на кнопку «Пошук». Кнопка «Рекомендації» впорядковує ще не оцінені користувачем альбоми за ступенем інтересу в порядку убування з урахуванням фільтрів або без, якщо вони не були виставлені.

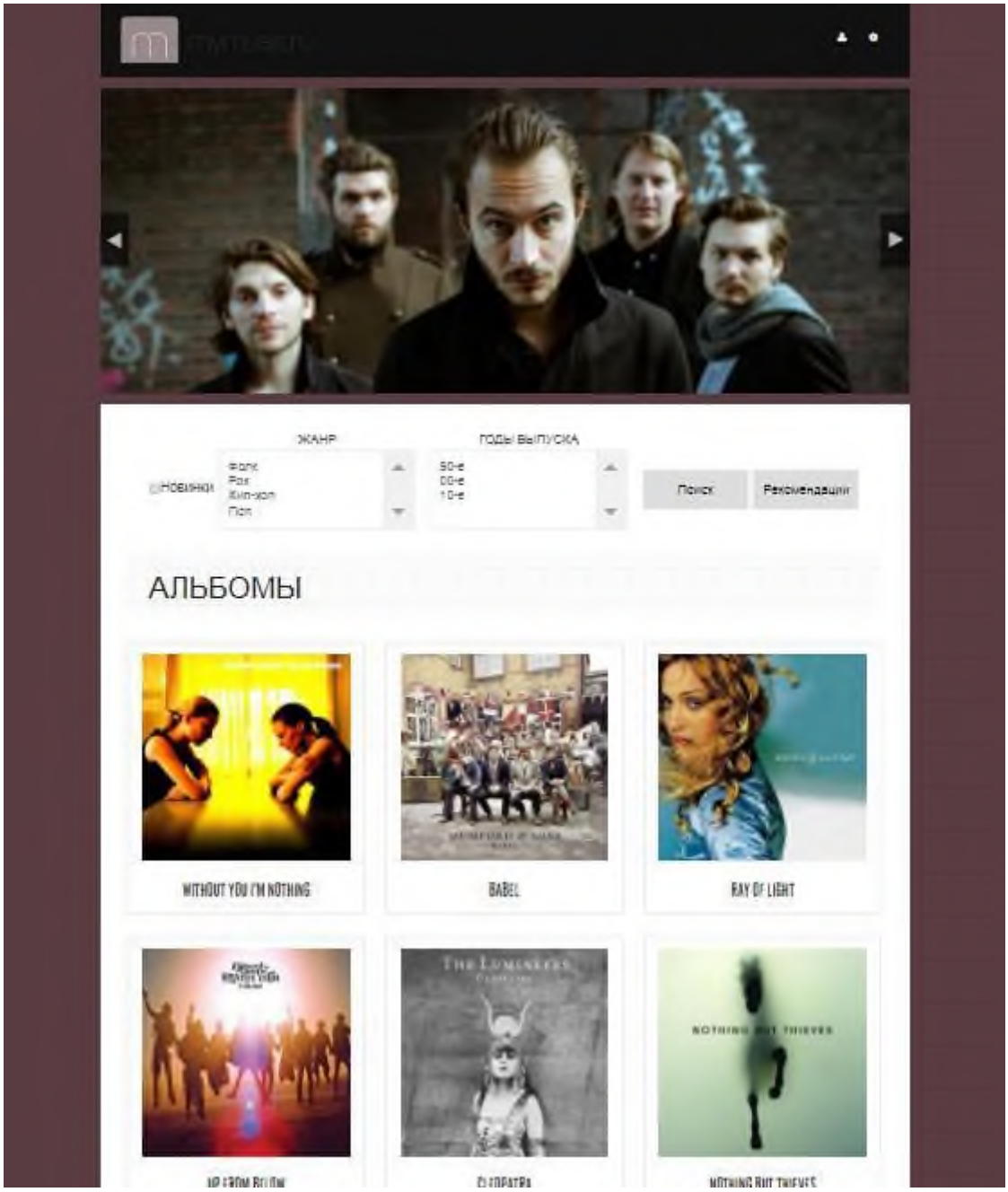

Рис. 3.3. Головна сторінка сайту

Каталог альбомів складається з усіх музичних альбомів, що містяться на сайті, які представлені у вигляді обкладинки альбому і його назви. Альбоми розташовані у вигляді стовпців і рядків: по три альбоми в рядку. Кількість рядків може бути необмежено і залежить від кількості альбомів, наявних в базі даних. При натисканні на будь-якої альбом користувач переходить на присвячену йому сторінку.

На сторінці альбому користувач може отримати інформацію про виконавця, прослухати альбом і, якщо користувач авторизований, поставити йому оцінку від одного до п'яти (рис. 3.4).

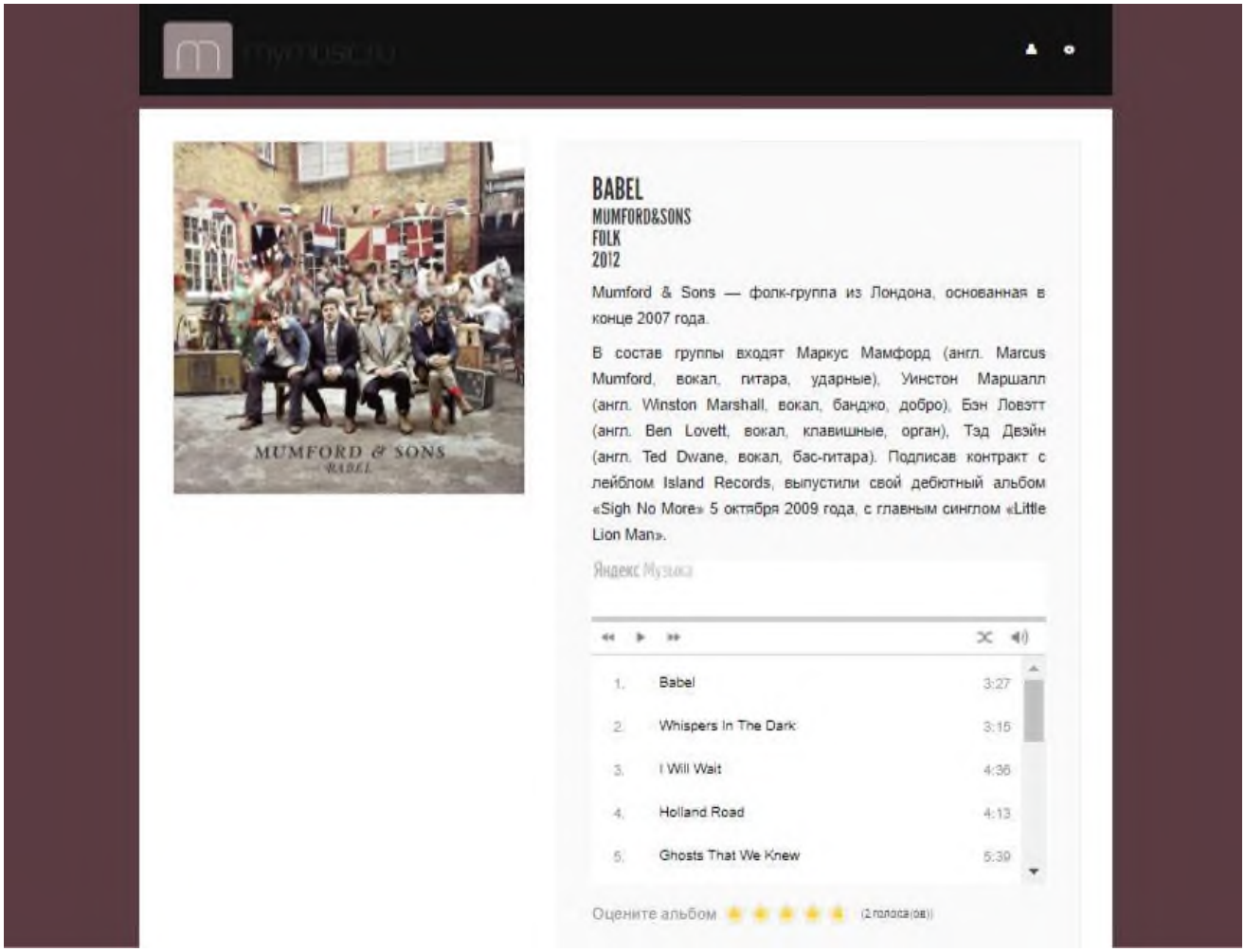

Рис. 3.4. Сторінка альбому

Щоб мати можливість виставляти оцінки і отримувати рекомендації, відвідувачеві сайту необхідно авторизуватись. Для цього потрібно клікнути по іконці «Увійти». Після цього користувачеві відкриється сторінка авторизації, на якій йому пропонується ввести свій логін і пароль (рис. 3.5). Для завершення процесу входу на сайт потрібно натиснути на кнопку «Увійти». Далі користувачеві відкривається сторінка, на якій міститься інформація про його профілі і кнопка «Вийти», натиснувши на яку можна вийти з профілю.

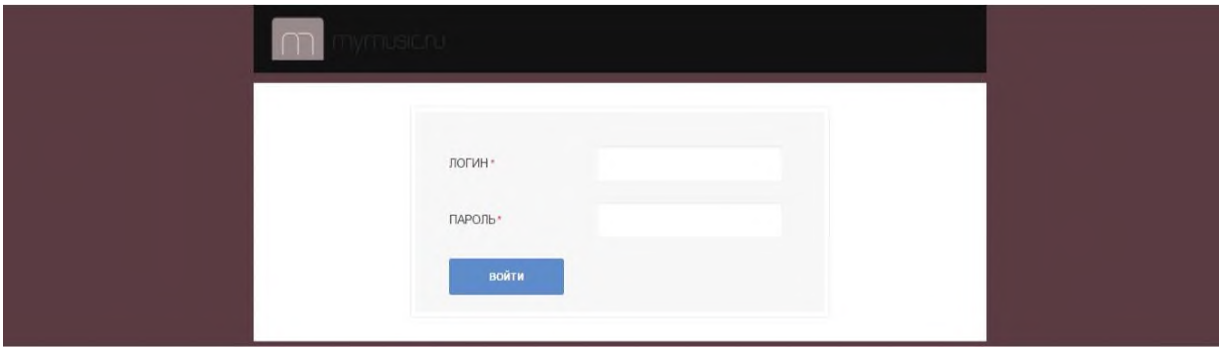

Рис. 3.5. Сторінка авторизації

Якщо у відвідувача сайту ще немає свого аккаунта, він може клікнути по іконці «Реєстрація». Після цього йому відкриється сторінка з формою реєстрації, де йому потрібно ввести свої ім'я, логін, пошту і пароль, а також підтвердити пошту і пароль (рис. 3.6). Для завершення реєстрації необхідно натиснути кнопку «Зареєструватися».

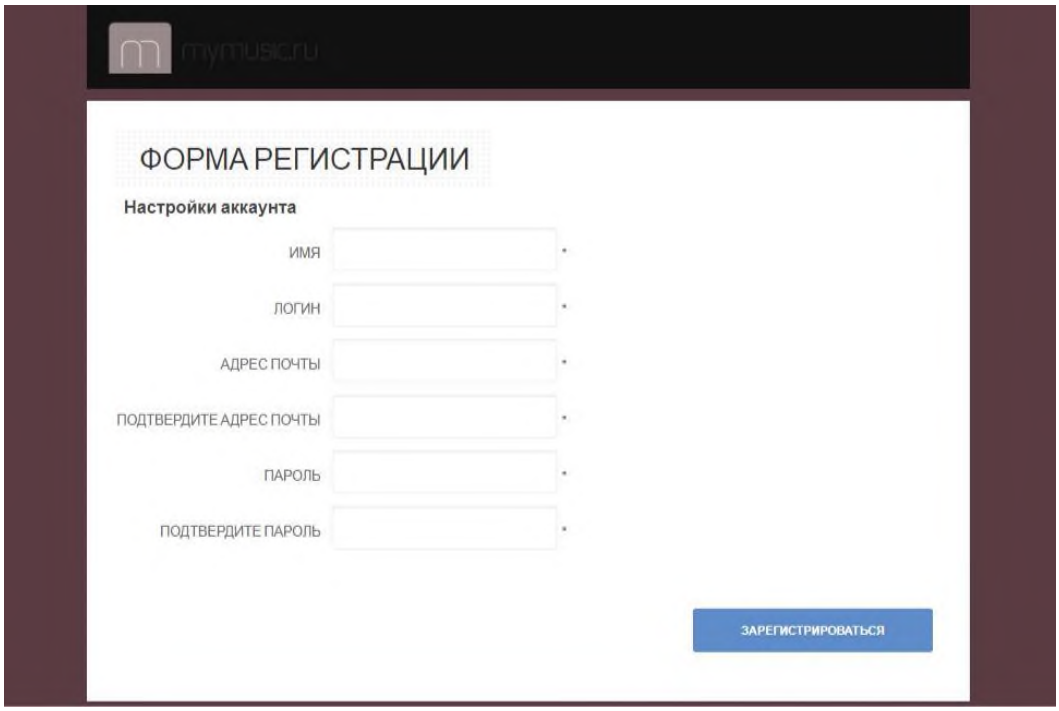

Рис. 3.7. Сторінка реєстрації

Якщо користувач не хоче переглядати всі наявні на сайті альбоми, він може вибрати альбоми певних жанрів і років випуску, а також новинки, в панелі фільтрації на головній сторінці. Після натискання кнопки «Пошук» сторінка перезавантажиться і виведе альбоми, що задовольняють запиту користувача.

На рисунку 3.8 представлений приклад фільтрації альбомів жанрів «фолк» і «рок», випущених в 2010 рік і пізніше.

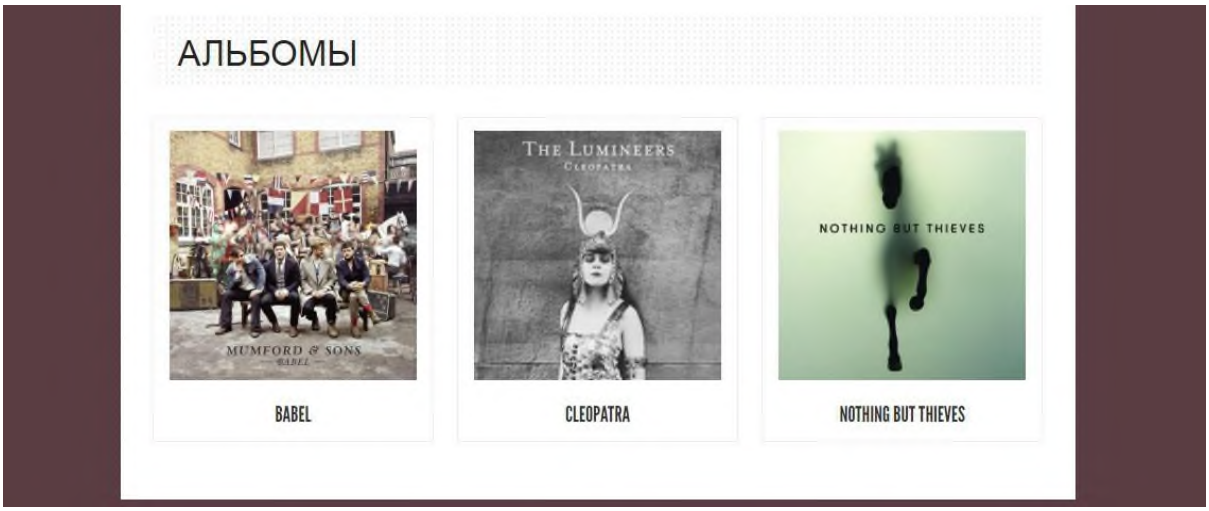

Рис. 3.8. Фільтрація альбомів

Якщо користувач авторизований, він може скористатися рекомендаційної системою, що надається сайтом. Для цього необхідно натиснути кнопку «Рекомендації», розташовану на головній сторінці, після чого в області відображення всіх альбомів, з'являться ті альбоми, які ще не оцінив, крім того, вони будуть розташовані в порядку убування: від більш цікавих до менш (рис. 3.9).

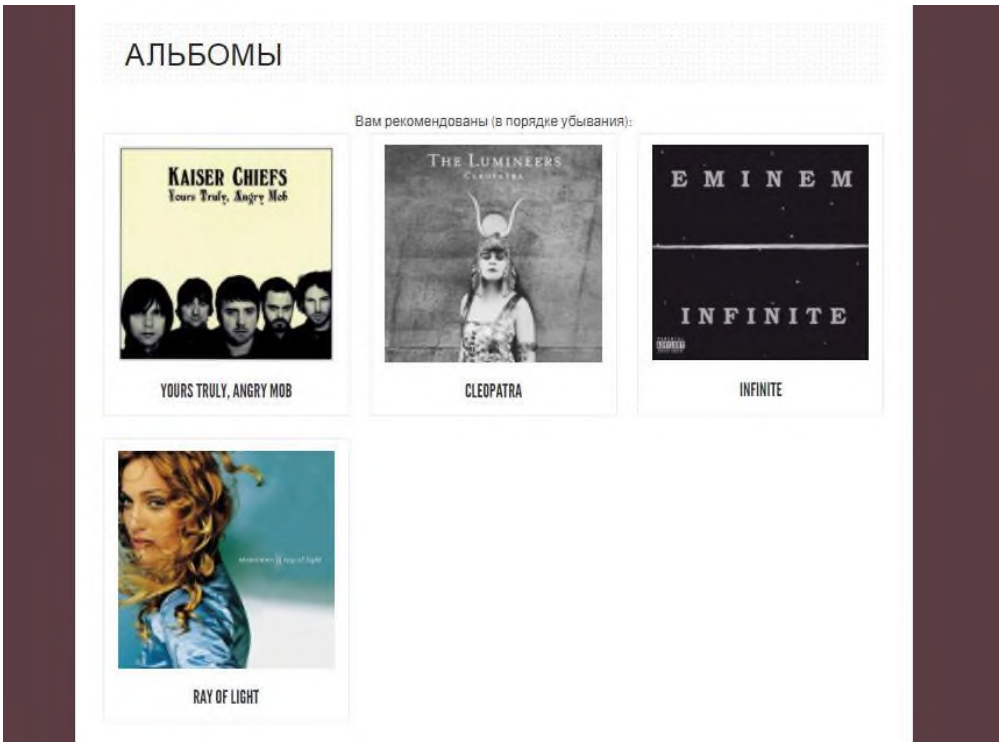

Рис. 3.9. Отримання рекомендацій

При обробці пошукового запиту на сервері відбувається підключення до бази даних і пошук у ній схожих найменувань композицій, після чого повертається знайдена запис, або повідомлення про відсутність такої.

При обробці даних грає композиції враховуються тільки частотні параметри і *id* композиції в базі даних, щоб виключити її з пошуку, так як вона, найімовірніше, виявиться найбільш підходящою. Повернено буде запис про композиції, у якій буде найменша різниця з поточної. Обробка даних відбувається під час програвання поточної композиції і припиняється з поверненням найбільш придатною знайденої запису при перемиканні композиції в результаті завершення програвання або натискання на кнопку «*Skip*».

Для отримання початкових значень частот була використана консольна утиліта *Sound eXchange* (*SoX*), призначена для роботи з аудіо-файлами, їх перетворення, конвертації, накладення ефектів, але для нас вона примітна тим, що з її допомогою можна отримати значення частот на протягом усієї музичної композиції, а також отримати метадані композиції.

Згодом, був написаний скрипт для адміністратора, що дозволяє одним рядком в консолі обробити композицію, обчисливши і додавши всі необхідні параметри в базу даних. Роботу починає скрипт на мові *bash*, який одержує через утиліту *SoX* метадані і частоти композиції, і поміщає дані в текстові файли для подальшого аналізу. Також, скрипт отримує на вхід текстовий файл, що містить текст композиції, він змінює його кодування для правильного відображення і поміщає файл в спеціальну директорію. Далі, необхідні параметри передаються скрипту на *Node.js*, і там відбувається їх обробка і додавання в базу даних сформованої записи. Спочатку зчитуються метадані і записуються в відповідний об'єкт скрипта, потім зчитуються частоти і відбувається їх перетворення в необхідні параметри, вони також записуються в об'єкт.

3.5. Висновки до розділу

У цьому розділі були описані дані, на яких проводились експерименти, потім були представлені заходи точності і повноти для оцінки роботи алгоритмів. Далі були коротко описані розроблені програми, код яких наведено в додатку. Нарешті були описані проводяться експерименти і розглянуті результати порівняльного аналізу методів рекомендацій *Slope One* і *RASP*.

Система рекомендацій прагне передбачити рейтинг або переваги, які користувач надав би товару, враховуючи його старі рейтинги або переваги.

Бібліотека глибокого навчання *FastAI* надає функціональність для легкого завантаження наших даних та побудови нашої спільної моделі фільтрації / рекомендації.

Для візуалізації роботи нейронної мережі розраховані рекомендації було завантажено на сайт з переглядом та прослуховуванням музикальних кліпів.

ВИСНОВКИ

З урахуванням постійно зростаючого обсягу інформації в Інтернеті системи, що рекомендують, стали ефективною стратегією подолання такого інформаційного перевантаження. Корисність систем, що рекомендують, неможливо переоцінити, враховуючи їх широке впровадження у багатьох веб-додатках, а також потенційний вплив на полегшення багатьох проблем, пов'язаних із перевибором вибору. За останні роки глибоке навчання викликало значний інтерес до багатьох наукових досліджень, таких як комп’ютерний зір та обробка природних мов, завдяки не лише зоряній продуктивності, але й аративній властивості вивчення представлення особливостей з нуля. Вплив глибокого навчання також широко поширений, нещодавно демонструючи свою ефективність у застосуванні до пошуку інформації та дослідження систем рекомендацій. Очевидно, що поле глибокого навчання в системі рекомендацій є поживним. Ця робота має на меті забезпечити всебічний огляд останніх дослідницьких досліджень систем рекомендацій на основі глибокого навчання. Більш конкретно, ми пропонуємо і розробляємо таксономію моделей рекомендацій на основі глибокого навчання, а також надаємо вичерпне резюме сучасного стану.

У даній роботі був запропонований метод формування рекомендацій на основі візерункових структур (*RAPS*). Незважаючи на те, що розроблений метод орієнтований на рекомендацію музичних композицій, його з легкістю можна використовувати і в інших областях, де користувачі оцінюють якісь об'єкти.

Проведені експерименти, які порівнюють методи *RAPS* і *Slope One*, показали, що рекомендаційна система на основі візерункових структур має досить непогані показники точності, повноти і час роботи.

Безумовно, в подальшому буде необхідно порівняти *RAPS* з іншими існуючими алгоритмами рекомендацій для того, щоб зробити більш повне висновок про його практичної застосовності.

У першому розділі були розглянуті основні визначення аналізу формальних понять і візерункових структур, які використовуються для класифікації груп

користувачів за їх вподобаннями. Також були розглянуті розширення візерункових структур для інтервалів і інтервальних векторів.

У другому розділі були розглянуті основні підходи до рекомендацій: коллаборативна фільтрація і фільтрація вмісту. Потім було розглянуто метод рекомендацій *Slope One,* який в подальшому буде використовуватися як базовий при проведенні експериментів. Був представлений розроблений підхід до формування рекомендацій на основі візерункових структур (*RASP*).

Додатково було проаналізовано приклади оцінки вподобань користувачів для музичних систем.

У третьому розділі були описані дані, на яких проводились експерименти, потім були представлені заходи точності і повноти для оцінки роботи алгоритмів. Далі були коротко описані розроблені програми, код яких наведено в додатку. Нарешті були описані проводяться експерименти і розглянуті результати порівняльного аналізу методів рекомендацій *Slope One* і *RASP*.

Система рекомендацій прагне передбачити рейтинг або переваги, які користувач надав би товару, враховуючи його старі рейтинги або переваги.

Бібліотека глибокого навчання *FastAI* надає функціональність для легкого завантаження наших даних та побудови нашої спільної моделі фільтрації / рекомендації.

Для візуалізації роботи нейронної мережі розраховані рекомендації було завантажено на сайт з переглядом та прослуховуванням музикальних кліпів.

Робота по даній темі буде продовжена за наступними напрямками:

подальше дослідження та доопрацювання методу *RAPS*;

дослідження і розробка другого методу формування рекомендацій на основі візерункових структур. Є припущення про те, що якщо брати другий оператор Галуа не від однієї музичної композиції, як це робиться в *RAPS*, а відразу від декількох музичних композицій, наприклад з високими оцінками, то можуть вийде хороші результати;

порівняння з моделями рекомендаційних систем *SVD* і *SVD* ++.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Ben Krose and Patrick van der Smagt. An introduction to Neural Networks*. (1996). – 126 *c*.

2. *Boucher-Ryan P, Bridge D.G., Collaborative Recommending using Formal Concept Analysis*. *Knowl.-Based Syst*. 19(5), 2006, *pp*. 309-315

3. *Ganter B., Kuznetsov S.O. Pattern structures and their projections, H.S. Delugach, G. Stumme* (*Eds*.), *ICCS, LNCS, vol*. 2120, *Springer*, 2001, *pp*. 129-142.

4. *Ganter B., Wille R. Formal Concept Analysis, mathematical foundations ed., Springer*, 1999.

5. *Kaytoue M., Kuznetsov S.O. Mining gene expression data with pattern structures in formal concept analysis*. *Inf. Sci*. 181(10), 2011, *pp*. 1989-2001.

6. *Koren Y., Robert M., Bell R.M. Volinsky C. Matrix Factorization Techniques for Recommender Systems*. *IEEE Computer* 42(8), 2009, *pp*. 42-49

7. *Lemire D., Maclachlan A. Slope One Predictors for Online Rating-Based Collaborative Filtering*. *SDM*, 2005, *pp*. 471-475.

8. *Melville P., Sindhwani V. Recommender Systems, Encyclopedia of Machine Learning, Claude Sammut and Geoffrey Webb* (*Eds*), *Springer*, 2010.

9. *Recommender Systems Handbook* / *F. Ricci, L. Rokach, B. Shapira* [*et al*.]. – *New-York* : *Springer Science+Bussiness Media*, 2011. – 842 *p*.

10. *Recommender systems*: *an introduction* / *D. Jannach, M. Zanker, A. Felferning* [*et al*.]. – *New-York* : *Cambridge University Press*, 2011. – 352 *p*.

11. Бьюли А. Изучаем *SQL* / Алан Бьюли. – Символ-Плюс, 2007. – 312 с.

12. Джонс, М. Рекомендательные системы: Часть 1. Введение в подходы и алгоритмы. – *URL*: *http://www.ibm.com/developerworks/ru/library/os-recom.html* (дата звертання: 01.12.2020).

13. Джонс, М. Рекомендательные системы: Часть 2. Механизмы с открытым

14. Жернакова, О. Системы рекомендаций и поиска видеоконтента. – Дата обновления : 01.02.2012. *URL* : *http://www.telemultimedia.ru/art.php?id=464.html* (дата звертання: 01.12.2020).

15. Рекомендационная система *IBM* с примерами исходных кодов. – *URL*: *http*://*www.ibm.com*/*developerworks*/*ru*/*library*/*os-recom*2.*html* (дата звертання: 01.12.2020).

16. Молинаро Энтони. *SQL*. Сборник рецептов / Энтони Молинаро. – *O'Reilly*, 2009. – 672 с.

17. Нейронні мережі, їх застосування та основні функції. Режим доступу: *https*://*clck.ru*/*GLKYq*, вільний.

18. Нечипорук В.В., Жмуденко І.Р. Нейронна система визначення користувацьких переваг// Тези доповідей наук.-практ. конф. “Сучасні тенденції розвитку системного програмування” (25-26 листопада 2020 р.). – К.: НАУ, 2020. – С. 19.

19. Штучний інтелект. Системи спілкування та експертні системи. Кн. 1 / Під ред. Э.В.Попова. – М.: Радіо та зв'язок, 1990. – 461 с.

20. ДСТУ 3008-95 Документація. Звіти у сфері науки і техніки. Структура і правила оформлення.

21. НД ТЗІ 1.1-003-99. Термінологія у області захисту інформації в комп'ютерних системах від несанкціонованого доступу. // Департамент спеціальних телекомунікаційних систем і захисту інформації Служби безпеки України. – Київ, 1999.

22. Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: НАУ, 2017. – 63 с.

Додаток А

Лістинг коду основного модулю програми побудови нейронної мережі

```
function
[recommendations,number_of_recommendations]=raps(items,train_users,user,left_border,
right_border,item_num)
global inf;
% %granici description
recommendations=[];
number_of_recommendations=zeros(item_num,1);

for k=1:size(items,1)

   item=items(k);
   if ((inf(user,items(k))>=left_border)&&(inf(user,items(k))<=right_border))
      dleft=zeros(item_num,1);
      dright=zeros(item_num,1);



      % %nahodim polzovateley, kotorie ocenili film 'item'
      set_users=[];
      temp_index=find(inf(:,item));
      for i=1:size(temp_index,1)
         if ( (isempty( find( train_users==temp_index(i) ) )==false) && ...

(inf(temp_index(i),item)>=left_border)&&(inf(temp_index(i),item)<=right_border) )
            set_users=[set_users; temp_index(i)];
         end
      end
      size(set_users);
```

```
set_users;


% % sozdaem description na osnove etih polzovateley
for i=1:size(set_users,1)
   temp_index=find(inf(set_users(i),:));

   for j=1:size(temp_index,2)
      item_i=temp_index(j);
      item_r=inf(set_users(i),temp_index(j));

      if ((dleft(item_i)==0)&&(dright(item_i)==0))
         dleft(item_i)=item_r;
         dright(item_i)=item_r;
      elseif (dleft(item_i)>item_r)
         dleft(item_i)=item_r;
      elseif ((dleft(item_i)<item_r)&&(dright(item_i)<item_r))
         dright(item_i)=item_r;
      end
   end
end

%dleft
%dright
% % vozvrashaem filmi, u kotorih granici >=left_border&&<=right_border

for i=1:item_num
   if ( (dleft(i)>=left_border)&&(dright(i)<=right_border) )
      number_of_recommendations(i)=number_of_recommendations(i)+1;
   end
end
```

```
    end
end
number_of_recommendations;



recommendations=find(number_of_recommendations);
for k=1:size(items,1)
    recommendations(recommendations==items(k))=[];
end



end
function
[recommendations,ratings]=slope_one(train_items,train_users,user,left_border,right_bord
er,min_border,max_border,item_num)

global inf;

ratings=zeros(item_num,1);
for i=1:size(train_items,1)
    ratings(train_items(i))=inf(user,train_items(i));
end
ratings;

for i=1:item_num

    if (ratings(i)==0)
        P=0;%prognoz
        R=0;
        for j=1:size(train_items,1)
```

```
        temp=(inf(train_users,i)).*inf(train_users,train_items(j));
        temp_index=find(temp);%ishem S_j,i(train_items)
        temp=inf(train_users,i)-inf(train_users,train_items(j));
        card=size(temp_index,1);
        if (card>0)
            dev=0;
            for k=1:card
                dev=dev+temp(temp_index(k));
            end
            dev=dev/card;
            P=P+dev+inf(user,train_items(j));
            R=R+1;
        end
    end
    if (R>=1)
        P=P/R;
        ratings(i)=P;
    end

  end

end

%ratings

recommendations=[];
for i=1:item_num
  r=ratings(i);
  if (r<min_border)
    r=min_border;
  end
```

```
    if (r>max_border)
        r=max_border;
    end
    if ((r>=left_border)&&(r<=right_border))
        recommendations=[recommendations; i];
    end
end
recommendations;

for k=1:size(train_items,1)

    recommendations(recommendations==train_items(k))=[];

end

end

function
[precision,recall]=precision_and_recall(user,test_items,retrieved_items,left_border,right_b
order)
global inf;

for i=size(retrieved_items,1):-1:1
    if (isempty( find(retrieved_items(i)==test_items) )==true)
        retrieved_items(i)=[];
    end
end
relevant_items=[];
for i=1:size(test_items,1)
    if ( (inf(user, test_items(i))>=left_border)&&(inf(user, test_items(i))<=right_border) )
        relevant_items=[relevant_items; test_items(i)];
```

```
    end
end
%retrieved_items
%relevant_items

if (isempty(relevant_items)==true)
    precision=0;
    recall=1;
elseif (isempty(retrieved_items)==true)
    precision=0;
    recall=0;
else
    temp=0;
    for i=1:size(retrieved_items,1)
        if ( isempty( find(relevant_items==retrieved_items(i)) )==false )
            temp=temp+1;
        end
    end

    precision=temp/size(retrieved_items,1);
    recall=temp/size(relevant_items,1);
end
end
```